64 PLUS 4
& AMIGA

MARZEC 1991 Indeks 377112 CENA 5000 zł

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE

W NUMERZE:

- Slang
- Ogonki i C-16
- D-Fast (Memory Controller)-AMIGA



W numerze :

W następnym numerze :

- C-64 i RESET
- Tworzymy własne DEMO
- D-MON PROFESSIONAL na AMIGĘ
- C-64 i sampling

OGŁOSZENIA

KOMPUTEROWA FIRMA USŁUGOWA "TREND" - COMMODORE AMIGA: oprogramowanie, literatura. Informacja : dyskietka lub koperta + znaczek. Kontakt: Rafał Wierzbicki, ul. Rogowska 86/10, 54-440 Wrocław.

AMIGA/C64/16/+4/ATARI-literatura, oprogramowanie, CARTRIDGE - Informator 4 znaczki, MIKROPOL, skr.1494, Łódź 37

Potrzebujesz jakiegoś programu na COMMODORE 64 (tylko magnetofon) - napisz do HAMMER'S THUNDER, ul. Okólna 24, 71-742 Szczecin

Kupię tanie dyskietki 5,25" i 3 1/2" C.Muszyński, 01-114 Warszawa, ul. Traktorzystki 4.

"LIFTER"
programy na C-64 (taśma i dysk), bogaty wybór, atrakcyjne ceny, szybkie terminy.
Katalogi - po załączeniu znaczka.
Jerzy "Lifter" Poprawa
ul. Przestrzenna 32/12
50-533 Wrocław

Wymienię programy na C-64 (dyskietki). Przemek Jaworski, skr.poczt.250, 58-500 Jelenia Góra 1.

Sprzedam rozszerzenie RAM 512 KB do Commodore 128. Cwojdziński, Bydgoszcz, tel. 348-38, od 9^{00}-14^{00}.

Klub Komputerowy Stodoła
AMIGA
- oferuje najlepsze stacje dysków 3,5" i 5,25"
- serwis sprzętu firmy Commodore
- literatura (także 64 plus 4)
- akcesoria itp.

Zapraszamy codziennie, oprócz sobót i niedziel w godzinach 11^{00} - 20^{00}
Warszawa ul. Batorego 10
tel. 25-60-31 wew. 35.
Giełdy komputerowe w Stodole, sobota od 10^{00} - 15^{00}

Programy: C-64 i ZX-SPECTRUM. Tanio!!! Dużo nowości!!! Katalog po przesłaniu koperty zwrotnej+znaczek. Marcin Eckerdorf, ul. Sikorskiego 121/11, 66-400 Gorzów Wlkp.

Serwis C-16/116/+4. Rozszerzenia pamięci. Bydgoszcz, tel. 714-102

Przedsiębiorstwo ABUK S-ka z o.o. oferuje państwu szybką i tanią obsługę reklamową. Ogłoszenia drobne od osób indywidualnych (do 10 słów) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo.
Reklamy ramkowe (minimalny format - 20 cm^2): 1cm^2 ogłoszenia-4500zł, cała strona - 2,5 mln zł; kolor - odpowiednio 100% drożej.
Ogłoszenia przyjmujemy za pośrednictwem poczty.
Nasz adres :
„64 plus 4"
85-166 Bydgoszcz 43
skrytka pocztowa 64

Treść ogłoszenia z określeniem formatu reklamy (ewentualnie zamówieniem koloru) prosimy nadsyłać listem poleconym wraz z odcinkiem wpłaty (za pomocą przekazu pieniężnego) na konto Przedsiębiorstwa ABUK Bank Polska Kasa Opieki SA Oddział w Bydgoszczy, konto nr : 5.09011-400522.7-136-11-111.0
Dołączenie do zamówienia odcinka wpłaty przyspieszy zamieszczenie reklamy.

64 PLUS 4

miesięcznik nr 3(5) marzec 1991
cena 1 egz.: 5000 zł



Wydawca:
ABUK Spółka z o. o.
Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.

Adres redakcji: Redakcja „64 plus 4" 85-166 Bydgoszcz 43 skrytka pocztowa 64
Redagują: Marcin Dudar, Sambor Kuźma, Paweł Sołtysiński, Waldemar Szczygieł (red. nacz.), Robert Turliński, Wojciech Wasilewski.
Okładka: Sławomir Karolczak.
Skład: ABUK
Druk: Prasowe Zakłady Graficzne, 85-009 Bydgoszcz, ul. Dworcowa 13
zam. 558/91

Drodzy Czytelnicy!

Nasza propozycja wydawania zestawów PUBLIC DOMAIN spotkała się z dużym uznaniem i zainteresowaniem czytelników. Świadczą o tym Wasze listy i zamówienia. Duża ilość tych ostatnich spowodowała opóźnienia w realizacji wysyłek - za co bardzo przepraszamy. Dokładamy wszelkich starań, aby nadrobić zaległości i w przyszłości uniknąć tego rodzaju sytuacji.

Dużym zainteresowaniem cieszy się również program VOICETRACKER na Commodore 64. Przy okazji wyjaśnijmy, iż w trakcie przygotowywania programu do dystrybucji autor dostarczył nam najnowszą wersję Voicetracker'a - V4.0. Postanowiliśmy więc wprowadzić do sprzedaży, zamiast zapowiadanej w styczniowym numerze wersji V3.0, tę najnowszą. Aktualnie - po pokonaniu trudności technicznych - wprowadzamy do dystrybucji wersję taśmową!

W naszych planach mamy jeszcze kilka ciekawych propozycji - dowiecie się o nich z lektury następnego numeru.

Jeśli nie dosięgną nas żadne złośliwości losu to następny numer naszego pisma powinien ukazać się w nowej szacie graficznej - trzymajcie kciuki! Pamiętajcie o tym szukając „64 plus 4" w kioskach Ruchu.

Otrzymujemy od Was wiele listów w których sygnalizujecie, że w wielu kioskach nie ma naszego pisma. Niestety jest to wynik bardzo złej pracy niektórych oddziałów Ruchu. Dopóki przedsiębiorstwo to się nie zreorganizuje (Ruch aktualnie jest w stanie likwidacji) jedyną formą systematycznego otrzymywania naszego pisma jest prenumerata.

Aktualna cena „64 plus 4" w prenumeracie wynosi 4600zł. Prenumeratę można zawierać na dowolny okres do końca 1991r. Wystarczy przemnożyć w/w cenę przez ilość miesięcy i tak wyliczoną kwotę wpłacić na konto: Bank PKO SA Bydgoszcz, konto nr: 5.09011-400522.7-136-11-111.0. Bardzo prosimy o CZYTELNE wypełnianie blankietów wpłat. Oczywiście w dalszym ciągu aktualna jest nasza oferta dla chętnych do prowadzenia indywidualnego kolportażu naszego pisma (szczegóły w poprzednich numerach).

Przypominamy, że zalegle numery „64 plus 4" można nabywać w warszawskiej księgarni „Elektronika" przy ulicy Mokotowskiej 51/53 oraz za pośrednictwem redakcji (zamówione numery prześlemy za zaliczeniem pocztowym).

Redakcja

Z DALEKA I Z BLISKA .....Z DALEKA I Z BLISKA .....
Z DALEKA I Z BLISKA .....Z DALEKA I Z BLISKA .....

AMIGA

- Duddie - człowiek uznany przez czytelników „Kebaba" za najlepszego kodera w Polsce stwierdził, że kończy z kodowaniem. Narazie jednak pracuje nad nową wersją swojego monitora „D-MON PROFESIONAL". Niedługo zamieścimy szczegółową informację na jego temat.

- Raf z grupy Katharsis nawiązał kontakt z magazynem NEWS-FLASH. W ostatnim 16 numerze autorzy zamieścili jego muzyczki określając je jako bardzo dobre. Gratulujemy!

- Grupa Toker Team wraz z Shadows organizują w lutym kolejne Copy Party. Postaramy się w następnych numerach zdać relację z tej imprezy.

- Do Polski dotarły już nowe wirusy o nazwach JACK 1 i JACK 2. Są dość niebezpieczne bo jeszcze nie dotarła szczepionka. Wirusy te linkują się do zbiorów „object" i można je znaleźć w pierwszym hunk'u. Jack 1 wydłuża zbiór o 2468 bajtów, a Jack 2 o 2400. Obiecujemy zamieścić dokładniejsze informacje i program zabijający żyjątko.

- Doszło do nas jakoby konkurent Kebab'a „Shadows disk news" zmarł śmiercią tragiczną! W ten sposób Kebab znowu został sam na placu boju?

- Commodore opracował nową stację dysków do Amigi. Na jednej dyskietce 3.5" aż 1.52 MB danych !!!. Nasi górą!!!

- AMIGA 1500 Personal Home Computer - nowa oferta komputera dla całej rodziny. Konstrukcja bazuje w całości na Amidze 2000. Oprócz wprowadzenia dwóch napędów dysków 3,5" (drugi napęd zamiast HDD) wbudowano jej na stałe najbardziej przydatne i popularne programy (wg. oceny specjalistów): procesor tekstów, bazę danych i arkusz kalkulacyjny Platinum Works, graficzny i animacyjny Deluxe Paint III, symulator lotu Their Finest Hour, doskonałe gry Populus i Sim City oraz Battle Chess. Komputer można oczywiście rozbudować do standardu XT lub AT. Amiga 1500 pracuje w oparciu o mikroprocesor Motoroli 68000 z zegarem 7.14 MHz.

C - 64

- Super Snapshot v5. Jeśli wszyscy znacie rewelacyjny Action Replay v5 to zapewne myślicie, że jest to najlepszy cartridge dla C-64. Otóż jest ponoć jeszcze lepszy ! Nazywa się Super Snapshot i oferuje między innymi:

– siedmiokrotne przyśpieszenie SAVE, piętnasto - lub 25 - krotne przyśpieszenie LOAD;
– generatory nieskończonej ilości „żyć", sprite killer;
– możliwość programowania klawiszy funkcyjnych;
– monitor do sprite'ów;
– monitor do znaków graficznych (fontów);
– monitor do sampli !!!
– nowy świetny nibbler — SUPER DISK SNAPSHOT
– SCREEN COPY, który nagrywa i ładuje ekrany w kilku formatach graficznych: na drukarce STAR LC10C robi wydruki w kolorze !!!
– poprawiony monitor języka maszynowego, który oczywiście nie uszkadza zawartości pamięci komputera;
– monitor do stacji dysków;
– kopier do zbiorów file'owych działający również z dwiema stacjami i także na 1581;
– kopier do całych dysków;
– program czytający zbiory sekwencyjne (SEQ);
– parametry do ponad 150 gier umożliwiające bezbolesne skopiowanie sobie oryginałów.

Czy cartridge ten pobije AR5? Trudno powiedzieć. Jego zaletą jest na pewno niska cena: 35 funtów.

Postaramy się zdobyć jeden "żyjący„ egzemplarz tego modułu i przeprowadzić wyczerpujący test na łamach "64+4„. Ciekawe kiedy polscy piraci hardware'owi zrobią nasze, rodzime kopie SS v5 ?

ADRESY, ADRESY...

Przedstawiamy adresy kilkunastu firm zajmujących się „produkcją" gier i innych programów komputerowych. Jeśli ktoś z Was napisał dobry program to może wysłać go na jeden (lub kilka...) z niżej podanych adresów i kto wie...? Może dostanie pracę? Firmy te również chętnie wysyłają reklamówki, znaczki, torby itp. Łapcie za pióra!

Accolade

Unit 17
Lombard Business Centre
50 Lombard Road
London, SW11 3 SU

Audiogenic

Winchester House
Canning Road
Wealdstone
Harrow HA3 7SJ

Domark

Ferry House
51-57 Lacy Road
Putney
London SW15 1PR

Electronic Arts

Langley Business Centre
11-49 Station Road
Langley
Nr Slough
Berks SL3 8YN

Elite

Anchor House
Anchor Road
Aldritch
Walsall WS9 8 PW

Gremlin

Alpha House
10 Carver Street
Sheffield S1 4FS

Hewson

56 Milton Park
Abingdon
Oxon OX14 4RX

Konami Software Club

Bank Building
Bank Street
Newton Abbat
Devon TQ12 2JL

MicroProse

Unit 1
Hampton Road Industrial
Estate
Tetbury
Glos. GL8 8LD

Mirrorsoft

Irwin House
Southwark Street
London SE1

Ocean

6 Central Street
Manchester M2 5NS

Palace

The Old Forge
Caledonian Road
London N1

Psygnosis

Unit 2
South Harrington Building
182 Sefton Street
Liverpool L3 4BQ

US Gold

Unit 10
Parkway Industrial Centre
Heneage Street
Birmingham B1 3HS

I na zakończenie adres który powinni znać wszyscy:

Commodore UK

Commodore House
The Switchback
Gardner Road
Maidenhead
Berks SL6 7XA

Wszystkie adresy dotyczą firm brytyjskich - więc po każdym z nich należy dopisać „ENGLAND". Otrzymania przesyłek życzy:

HI-Man

MAPA PAMIĘCI - CZĘŚĆ IV

Kontynuując opis szczegółowej mapy pamięci komputerów rodziny C16, C116, Plus4 prezentujemy obecnie obszar od komórki o adresie 768 do komórki o adresie 1267 włącznie. Znajduje się tu tablica wektorów systemu operacyjnego, obszar roboczy (bufor) procedur współpracy z magnetofonem, kilka podprogramów systemu operacyjnego oraz - podobnie jak w omówionych wcześniej obszarach - szereg komórek roboczych i zmiennych systemu operacyjnego.

Znajdująca się w obszarze o adresach 768 do 817 tablica wektorów zawiera adresy procedur realizujących poszczególne funkcje systemu operacyjnego. Tablica jest zorganizowana w ten sposób, że każdy z adresów zajmuje dwie kolejne komórki pamięci. Przy wywołaniu tych procedur wykonywany jest skok pod adres pobrany z tablicy. Dzięki takiemu rozwiązaniu można łatwo modyfikować wykonywanie pewnych funkcji komputera poprzez napisanie własnej procedury i umieszczenie jej adresu początkowego w tablicy wektorów. Możliwe jest również modyfikowanie oprogramowania zawartego w pamięci ROM bez ryzyka, że programy nie będą pracowały poprawnie na wszystkich egzemplarzach komputera. Jeżeli po modyfikacji ulegnie zmianie adres startowy procedury ,wystarczy go wpisać na odpowiednie miejsce w tablicy wektorów. Programy wykorzystujące tę procedurę będą pracowały poprawnie, o ile będą pobierały jej adres z tablicy.

Jako przykład wykorzystania tablicy wektorów może posłużyć często stosowana do prostego zabezpieczania programów w BASICU zmiana wektora procedury listowania tekstu programu. Osiąga się ją poprzez umieszczenie za pomocą instrukcji POKE nowych wartości w komórkach o adresach 774/775. To, co komputer zrobi podczas wykonywania instrukcji LIST zależy od procedury, której adres zostanie tu umieszczony.

Przdstawiona poniżej tablica zawiera kilka przykładów praktycznych:

Kmórka 774	Komórka 775	Wektor IQPLOP	Działanie LIST
110	139	35694	normalnie
110	200	51310	listowane są tylko numery linii
129	134	34433	powoduje komunikat OUT OF MEMORY
0	128	32768	Powoduje „zimny stzrt" BASIC

Standardowo wektor IQPLOP wskazuje na początek zawartej w ROM procedury zamieniającej jednobajtowe kody słów kluczowych BASIC (token) w ciąg znaków ASCII. W przypadkach 2 i 3 nowe wektory powodują skok do wnętrza procedur ROM realizujących odmienne funkcje. W przypadku 4 wektor ten wskazuje na początek interpretera BASIC, i wykonanie LIST powoduje „zimny start" i zniszczenie tekstu programu

Przy rozbudowie interpretera BASIC o nowe rozkazy modyfikowany jest wektor IGONE zajmujący komórki 776/777. Po modyfikacji wskazuje on na procedurę, która rozpoznaje i wykonuje nowy rozkaz lub przekazuje sterowanie do procedur ROM, jeżeli jest to któryś z rozkazów standardowych.

Obszar pamięci o adresach 818 - 1010 wykorzystywany jest podczas współpracy z magnetofonem. Po wczytaniu programu z taśmy z komórek 818 - 838 można odczytać typ zbioru, adresy początku i końca ładowania oraz jego nazwę.

W obszarach o adresach 1139 - 1156 oraz 1172 - 1185 znajduje się grupa podprogramów wykorzystywanych podczas operacji wymagających przełączania banków pamięci.

W komórkach 1280 - 1282 zawarty jest kod rozkazu oraz adres skoku, który jest wykonywany podczas realizowania funkcji USR (x) BASIC. Wpisując do komórek 1281/1282 adres własnej procedury spowodujemy, że będzie ona realizowana podczas wykonywania tej instrukcji.

Podczas pierwszego startu komputera po włączeniu zasilania procedura inicjująca sprawdza wielkość pamięci RAM i ustawia wskaźniki DEJAVU, MEMSTR i MEMSIZ (komórki 1288 , 1329 - 1332). Wskaźniki MEMSTR i MEMSIZ zawierają adresy odpowiednio początku i końca pamięci RAM dostępnej dla systemu operacyjnego. Adresy te zależą od rozmiaru pamięci , jaka jest zamontowana w komputerze. Przy restartach systemu po użyciu klawisza RESET (przy wciśniętym klawiszu RUN/STOP) testowany jest wskaźnik DEJAVU i jeśli jest on ustawiony, to sprawdzanie pamięci nie jest wykonywane.

Zawartość komórki pamięci o adresie 1344 kontroluje funkcję samopowtarzania klawiszy przy dłuższym wciśnięciu.

Jeśli wpiszemy:

```
20 POKE 1344,64
```

to funkcja samopowtarzania zostanie wyłączona.

Odczytując zawartość komórki o adresie 1347 możemy sprawdzić stan klawiszy SHIFT, C= oraz CONTROL. Poszczególnym klawiszom odpowiadają następujące bity tego wskaźnika: bit 1 - SHIFT, bit 2 - C=, bit 3 - CONTROL.

Jeżeli wykonamy program (przerwać go można klawiszem RUN/STOP):

```
10 PRINT PEEK (1347): GOTO 10
```

i będziemy wciskać klawisze SHIFT, C=, i CONTROL to uzyskamy następujące wartości:

0	żaden	4	CONTROL
1	SHIFT	5	SHIFT+CONTROL
2	C=	6	C= + CONTROL
3	SHIFT + C=	7	SHIFT+C=+CONTROL

Wskaźnik MODE (adres 1351) kontroluje działanie klawisza SHIFT.

Jeżeli wpiszemy:

```
30 POKE 1351,128
```

to komputer nie będzie reagował na kombinację klawiszy SHIFT + C= , która normalnie przełącza generatory znaków powodując zmianę wyświetlanych na ekranie znaków z zestawu duże litery/znaki graficzne na zestaw małe/duże litery. Normalne działanie można przywrócić wpisując POKE 1351,0

W komórkach o adresach 1362 - 1368 przechowywana jest zawartość rejestrów procesora 7501 w momencie przejścia do monitora TEDMON po napotkaniu rozkazu BRK.

ADRES HEX	ADRES DEC	ETYKIETA	OPIS
0300-0331	768-817		wektory systemu operacyjnego
0300-0301	768-769	IERROR	komunikat błędu BASIC $8686
0302-0303	770-771	IMAIN	start BASIC$8712
0304-0305	772-773	ICRNCH	generowanie token $8956
0306-0307	774-775	IQPLOP	listowanie tekstu $8B6E
0308-0309	776-777	IGONE	wykonanie rozkazu $8BD6
030A-030B	778-779	IEVAL	ocena token $9417
030C-030D	780-781	IESCLK	generowanie user-token $896A
030E-030F	782-783	IESCPR	tworzenie token'u $8B88
0310-0311	784-785	IESCEX	przetwarzać user-token $8C8B
0312-0313	786-787	ITIME	przerwanie (zegar) $CE42
0314-0315	788-789	CINV	przerwanie sprzętowe $CE0E
0316-0317	790-791	CBINV	przerwanie BRK $F44C
0318-0319	792-793	IOPEN	procedury KERNAL OPEN $EF53
031A-031B	794-795	ICLOSE	CLOSE $EE5D
031C-031D	796-797	ICHKIN	CHKIN $ED18
031E-031F	798-799	ICKOUT	CHKOUT $ED60
0320-0321	800-801	ICLRCH	CLRCHN $EF0C
0322-0323	802-803	IBASIN	CHRIN $EBE8
0324-0325	804-805	IBSOUT	CHROUT $EC4B
0326-0327	806-807	ISTOP	STOP $F265
0328-0329	808-809	IGETIN	GETIN $EBD9
032A-032B	810-811	ICLALL	CLALL $EF08
032C-032D	812-813	USRCM	monitor-break $F44C
032E-032F	814-815	ILOAD	procedura KERNAL LOAD $F04A
0330-0331	816-817	ISAVE	SAVE $F1A4
0332-03F2	818-1010	TAPBUF	bufor operacji we/wy dla magnetofonu
0332	818		typ zbioru (0-BASIC, 1-pr. maszynowy)
0333-0334	819-820		adres początku bloku
0335-0336	821-822		adres końca bloku
0337-0346	823-838		nazwa programu (16 znaków)
03F3-03F4	1011-1012	WRLEN	licznik danych (write)
03F5-03F6	1013-1014	RDCNT	licznik danych (read)
03f7-0436	1015-1078	INPQUE	RS 232 bufor wejściowy (24 bajty)
0437-0472	1079-1138		obszar wykorzystywany podczas odczytu z taśmy do zaznaczania błędnie wczytanych bajtów
0473-0478	1139-1144	CHRGET	podprogram CHRGET
0479-0484	1145-1156	CHRGOT	podprogram CHRGOT
0485-0493	1157-1171	QNUM	
0494-04A1	1172-1185	INDSUB	podprogram wykorzystywany podczas operacji związanych z bankowaniem pamięci
04A2-04A4	1186-1188	ZERO	stała numeryczna dla BASIC
04A5-04E6	1189-1254		podprogramy bankowania
04E7	1255	PUFILL	znak zapełnienia przy PRINT USING
o4E8	1256	PUCOMA	symbol przecinka
04E9	1257	PUDOT	symbol kropki
04EA	1258	PUMONY	znak waluty
04EB-04EE	1259-1262	TMPDES	pamięć chwilowa dla INSTR
04EF	1263	ERRNUM	ostatni numer błędu
04F0-04F1	1264-1265	ERRLIN	numer wiersza w którym wystąpił ostatni błąd ($FFFF-nie było błędu)
04F2-04F3	1266-1267	TRAPNO	numer wiersza dla ON ERROR GOTO

Wojciech Wasilewski

P.S. Przewidujemy jeszcze trzy-cztery odcinki opisu mapy pamięci. Następnie zamierzamy opublikować cykl artykułów opisujących interesujące POKE'i . Wszystkich czytelników zapraszamy do współpracy przy realizacji tego cyklu.

Redakcja

Polskie litery dla C16/116/+4

Zamieszczona procedura może być używana w programach w BASIC'u dla uzyskania na ekranie polskich liter.

W liniach 1010-1060 zapisywana jest do obszaru pamięci $606-623 (dziesiętnie 1540-1571) i wykonywana procedura w języku maszynowym przypisująca standardową tablicę znaków z pamięci ROM (pod adresem $D000) do pamięci RAM pod adres $3C00. Linie 1070-1080 przedefiniowują niektóre znaki w tej tablicy, wpisując zamiast nich wzorce polskich liter zawarte w danych w liniach 1090-1170. Instrukcja w linii 1180 powoduje przełączenie sterownika graficznego TED na nową tablicę znaków.

Program w BASIC'u wykorzystujący tę procedurę powinien najpierw zarezerwować obszar pamięci $3C00-4000, w którym znajdzie się tablica znaków, przez ustalenie górnej granicy pamięci dostępnej dla interpretera BASIC'a na $3C00 (dziesiętnie 15360):

```
POKE 55,0 : POKE 56,60 :CLR
```

Następnie, wywołanie GOSUB 1000 zdefiniuje w wymienionym obszarze tablicę znaków zawierającą polskie litery na miejscu odpowiednich liter łacińskich wciskanych z klawiszem SHIFT: Ą = SHIFT-A, Ć = SHIFT-C itd. Kłopot z literami Ż i Ź został rozwiązany w ten sposób, że litera Ź kodowana jest na klawiszu SHIFT-Z, a litera Ż na klawiszu C=-Z. Sposób kodowania można oczywiście dowolnie zmienić: pierwsza liczba w każdej linii DATA jest kodem ekranowym znaku, który będzie zastąpiony polską literą, kolejno: Ą,Ć,Ę,Ł,Ń,Ó,Ś,Ź i Ż. Możemy w to miejsce wpisać własne kody, którym chcemy przypisać polskie litery.

Powrót do normalnej tablicy znaków uzyskać możemy przez GOSUB 2000, a powtórne włączenie tablicy z polskimi literami - GOSUB 1180.

Jeżeli procedura w języku maszynowym wykorzystywana do przepisania tablicy znaków kolidowała by z innymi programami zajmującymi obszar pamięci $606-623, można przenieść ją w inne miejsce przez zmianę adresów wpisanych w linii 1020 - procedura jest relokowalna.

Osoby posiadające komputer z pamięcią 64 kB uznają zapewne za marnotrawstwo ograniczenie pamięci dostępnej dla BASIC'a do zaledwie 11 kB. W tej sytuacji są dwa rozwiązania:

1. Umieścić program w BASIC'u w obszarze pamięci powyżej $4000 (podobnie jak przy korzystaniu z grafiki wysokorozdzielczej), wykonując zamiast podanych powyżej POKE'ów instrukcję SYS 50748.
2. Zmienić adres, pod którym będzie umieszczana w pamięci nowa tablica znaków. Zmodyfikowanie podanej procedury, tak aby tablica znaków znajdowała się pod innym adresem nie jest trudne, ale byłoby nieco skomplikowane w opisie, dlatego też przeróbkę tę zostawiam dociekliwym użytkownikom.

UWAGA: Gdy mamy aktywną własną tablicę znaków w RAM, po każdym komunikacie błędów w BASIC'u, wyjściu z monitora języka maszynowego do BASIC'a lub wyjściu z trybu graficznego do tekstowego (GRAPHIC0) zawartość komórki 65298 „psuje się" w ten sposób, że na ekranie pojawia się nieczytelna „kasza". W takiej sytuacji, aby przywrócić normalny obraz na monitorze należy wykonać instrukcję (niestety trzeba ją wpisać „na ślepo"):

```
POKE 65298,192
```

```
1000 REM POLSKIE LITERY
1010 RESTORE1030
1020 FORI=1540TO1571:READP$:POKEI,DEC(P$)
1025 NEXT:SYS1540
1030 DATAA2,00,BD,00,D0,9D,00,3C
1040 DATABD,00,01,9D,00,3D,BD,00
1050 DATAD2,9D,00,3E,BD,00,D3,9D
1060 DATA00,3F,E8,D0,E5,60,00,00
1070 FORZ=1TO9:READS:AD=DEC("3C00")+8*S
1080 FORI=0TO7:READP$:POKEAD+I,DEC(P$)
1085 NEXTI,Z
1090 DATA65,18,3C,66,7E,66,66,66,03
1100 DATA67,30,3C,66,60,60,66,3C,00
1110 DATA69,7E,60,60,78,60,60,7E,06
1120 DATA76,60,6C,78,70,60,60,7E,00
1130 DATA78,30,66,76,7E,7E,6E,66,00
1140 DATA79,30,3C,66,66,66,66,3C,00
1150 DATA83,0C,3C,60,3C,06,66,3C,00
1160 DATA90,0C,7E,0C,18,30,60,7E,00
1170 DATA109,7E,06,0C,7E,30,60,7E,00
1180 POKE65298,192:POKE65299,60
1190 RETURN
2000 REM NORMAL
2010 POKE65298,196:POKE65299,209
2020 RETURN
```

Listing: POLSKIE LITERY

Jarosław Rafa

Czas komputerów

Zapraszamy do podróży w czasie, w której dowiecie się jak to szare pudełko, które tak wielbicie i nazywacie komputerem zrodziło się w wyobraźni konstruktorów.

Początki

Od kiedy człowiek odczuł potrzebę liczenia, zawsze starał się ten proces uprościć. Pierwsze co odkrył to metoda liczenia na palcach. Wtedy też najpewniej powstał system dziesiętny, który nęka wszystkich koderów i programistów do tej pory. Gdyby tak człowiek miał szesnaście palców!? Zapewne dziś spokojnie obliczali byśmy sumę $1A+$D7 i nie szukali pomocy w kalkulatorach.

Człowiek stawał się coraz bardziej cywilizowany i zaczął używać coraz bardziej wyrafinowanych metod liczenia. Palce poszły w odstawkę i wynaleziono liczydło. Nikt nie jest pewny kiedy dokładnie to się stało (prawdopodobnie około roku 3000 p.n.e), ale jedno jest pewne : ten wynalazek przetrwał wieki i nadal jest w użyciu! W Japonii odbywają się do tej pory mistrzostwa rachmistrzów, którzy na liczydłach potrafią pracować kilkakrotnie szybciej niż na kalkulatorach.

Po wynalezieniu liczydeł przez wiele wieków nic ciekawego się nie działo. Uczeni jednak opracowywali podstawy matematyki i wymyślali teoretyczne urządzenia liczące.

Pierwsza maszyna

Blaise Pascal chcąc dopomóc swojemu ojcu, który był poborcą podatkowym, wynalazł pierwszą na świecie maszynę liczącą - Pascalinę. Zdolna była ona do dodawania i odejmowania, a do wyliczenia sumy używane były specjalne koła zębate. Niestety maszyna była droga, a ludzie niezbyt ufali jej wyliczeniom (chociaż były dobre), więc nie weszła ona do powszechnego użytku. Mimo tego był to pierwszy krok ku prawdziwej mechanizacji (czyt. komputeryzacji) procesu liczenia.

Kolejną postacią po Pascal'u był Gotfried Leibniz.

Na podstawie wynalazku Pascal'a skonstruował on maszynę, która potrafiła oprócz dodawania i odejmowania mnożyć i dzielić! Zawierała ona zębatkę pracującą na zasadzie krokowej! Był to wielki krok naprzód (nie zębatki oczywiście...). Maszyna była wyposażona w rączkę, która była przekręcana do przodu (dla mnożenia) lub do tyłu (dla dzielenia). Niestety i ta maszyna nie odniosła sukcesu lecz stało się tak tylko dlatego, że sam konstruktor stracił nią zainteresowanie.

Kolejny krok do przodu uczynił matematyk George Boole.

Dla wszystkich, którzy się nie domyślili: to on właśnie opracował algebrę Boole'a, która bardziej zajmuje się symbolami niż numerycznymi odpowiednikami.

Ojciec komputerów.

Następnym człowiekiem odnotowanym przez historię był Charles Babbage zwany również ojcem komputerów. Wynalazł on coś co nazwał „maszyną różnicową". Była ona jednak bardziej wymyślona niż zrealizowana. Rząd skasował dotacje dla Babbage'a i przez to nigdy nie była zrobiona do końca. Babbage nie poddał się jednak i zaczął pracować nad lepszą i większą. Wynalazł tzw. „maszynę analityczną" która jest już praprzodkiem Waszych cudów współczesnej elektroniki leżących na biurkach. Mogła ona przeprowadzać każdy typ operacji matematycznych, przechowywać program i podawać wyniki operacji.

W 1890 roku Herman Hollerith opracował specjalną maszynę dla przyśpieszenia spisu ludności. Nazwana była Tabulatorem Hollerith'a i po raz pierwszy wykorzystywała karty dziurkowane w przetwarzaniu informacji. Hollerith założył potem firmę Tabulating Machine Company, która z czasem przekształciła się w dzisiejszego potentata: IBM.

Kolejnym krokiem była praca Alan'a Turing'a „On Computable Numbers" opublikowana w roku 1936. Została ona uznana za jedną z najważniejszych prac naukowych z dziedziny komputerów. Turing brał również udział przy konstruowaniu komputera „Colossus 1", który był używany do łamania kodów w czasie wojny.

Pierwszą elektromechaniczną maszyną liczącą był Harvard Mark 1 opracowany na Uniwersytecie a Harwardzie. Była to maszyna bardzo droga, tak że prawdopodobnie nigdy by nie powstała gdyby nie pomoc firmy IBM. Komputer ten potrafił dodać dwie liczby w trzy dziesiąte sekundy! W tych czasach był to bardzo dobry wynik.

Tak zakończyła się era komputerów zerowej generacji.

Pierwsza generacja.

Pierwszą maszyną tej generacji był ENIAC (Electronic Numerical Integrator And Calculator) skonstruowany po trzech latach pracy przez I.P.Eckert'a i I.W. Mauchły'ego. Zajmował dość sporą halę o wymiarach 15x10 m. Pobierał moc 174 kW, zawierał 17468 lamp, 1500 przekaźników, 70000 oporników i 10000 kondensatorów. Był jednak bardzo szybki (oczywiście w porównaniu do maszyn 0 - generacji). Czas dodawania wynosił 0,2 ms, a mnożenia 2,8 ms.

Przełomowym dniem w historii komputerów był 23 grudnia 1947 roku. Wtedy to John Bardeen, William Shockley i Walter Brattain opracowują pierwszy tranzystor.

W 1948 roku John von Neumann proponuje architekturę komputerów stosowaną do tej pory. W skrócie polega ona na zapamiętywaniu programu i danych w tej samej pamięci.

Komputerowy boom.

Począwszy od roku 1951 postęp następuje bardzo szybko. Powstaje następca ENIAC'a - UNIVAC 1. Jest to pierwszy sukces handlowy. W sumie zainstalowano 75 systemów UNIVAC. Zegar pracował z częstotliwością 2,25 MHz, czas dodawania 120 us, a mnożenia 1,8 ms. Komputer ten po raz pierwszy wykorzystywał taśmę magnetyczną do przechowywania danych.

Dwa lata później zostaje opracowana pamięć na rdzeniach ferrytowych. Ziller i Backus wymyślają język FORTRAN (FORmula TRANslator).

W 1955 roku zaczyna się era komputerów drugiej generacji. Dopiero teraz masowo zaczynają być stosowane tranzystory. Bell Telephone Laboratories wypuszcza komputer wykorzystujący ok. 800 tranzystorów i ok. 11 tys. diod germanowych.

Zostaje skonstruowana pierwsza pamięć magnetyczna. Była bardzo mała i bardzo droga, ale była!

Firmy IBM i Texas Instruments właściwie nadają „pędu" postępowi. Jack Kilby konstruktor TI opracowuje w 1958 r. pierwszy układ scalony, a już w 1961 roku ta sama firma produkuje pierwszy komputer całkowicie oparty na „scalakach" z pamięcią półprzewodnikową!

W 1974 roku INTEL rozpoczyna produkcję pierwszego mikroprocesora 8080. Trzy lata później, w 1977 roku na arenę zmagań z postępem wkraczają nasi, tzn. firma Commodore. Wypuszcza ona pierwszy komputer osobisty PET (Personal Electronic Transactor). Dalej już komputerowy boom następuje bardzo szybko. IBM w 1981 roku prezentuje PCety. Zaczynają też być produkowane komputery domowe - ZX Spectrum, Commodore 64, Atari 800XL. W 1984 roku powstaje też pierwsza Amiga. Komputery stają się coraz mniejsze i coraz szybsze. Do czego dąży technika?

Obecnie pracuje się nad komputerami opartymi o nadprzewodniki, światło czy - już najbardziej wyrafinowane - neutrony. Jak daleko zajdziemy ?

Podróż odbyłem korzystając z własnego pamiętnika, CHIP'a i Commodore Disk User.

Sambor Kuźma

Graj aby wygrać

Przedstawiamy kolejną porcję „chwytów" do gier dla C-64. Jeśli któraś z poprawek nie będzie działała to znaczy, że masz inną wersję gry niż autor tej rubryki lub po prostu coś źle zrobiłeś.

Większość poprawek wpisuje się wg następującego schematu:

> 1. Wczytanie i uruchomienie gry.
> 2. Wciśnięcie przycisku „RESET" (konieczny jakikolwiek cartridge lub własny przycisk - w następnym numerze opiszemy jak go zainstalować).
> 3. Wpisanie podanych poprawek.
> 4. Uruchomienie gry w podany przez nas sposób.

Jeśli któraś z poprawek będzie wymagała wykonania innych operacji - zostanie to szczegółowo omówione. A teraz zaczynamy:

MEGA APOCALYPSE

POKE 32417,173
POKE 32509,173 - nieśmiertelność
SYS 22562 - uruchomienie gry

URIDIUM

POKE 3452,0 - nieśmiertelność
POKE 13468,nr
POKE 13469,nr - nr poziomu startu gry
POKE 13467,il - ilość żyć
SYS 2461 - uruchomienie gry

BOMB JACK I

POKE 4056,173 - nieśmiertelność
SYS 2238 - uruchomienie gry

BOMB JACK II

POKE 10715,234
POKE 10716,234
POKE 10717,234 - nieśmiertelność
SYS 15146 - uruchomienie gry

FEUD

POKE 16404,15
POKE 17204,15 - już masz wszystkie składniki
POKE 17591,1-40 - szybkość
SYS 16384 - uruchomienie gry

TIGER MISSION

W czasie wyświetlania ekranu tytułowego należy jednocześnie nacisnąć klawisze: CTRL,C=,2,Q,L,I,K. Na dole ekranu pojawi się napis: „CHEAT MODE" i wykonanie wszystkich misji jest o wiele łatwiejsze.

INTO THE EAGLE'S NEST

POKE 24651,234
POKE 24652,234
POKE 24653,234 - nieskończona ilość żyć i nieruchomi przeciwnicy
SYS 32784 - uruchomienie gry

KRACKOUT

POKE 44388,234
POKE 44389,234
POKE 44390,234 - nieśmiertelność
POKE 32934,0-100 - poziom startowy
SYS 32837 - uruchomienie gry

MUTANTS

POKE 9273,234
POKE 9274,234 - nieśmiertelność
SYS 4096 - uruchomienie gry

OLLI & LISSA

POKE 8513,0 - wyłączona kolizja obiektów
SYS 7424 - uruchomienie gry

Zachęcamy do robienia własnych przeróbek. Przysyłajcie je do nas, a my będziemy je publikować. Planujemy również zorganizowanie konkursu na „najlepszego przerabiacza gier". Co o tym myślicie? Czekamy na Wasze opinie.

Hi-Man

Trochę o zmiennych systemowych

Tytuł artykułu jest może co nieco niejasny, ale dla tych, którzy się z zagadnieniem tzw. „mapy pamięci" zetknęli rozszyfrowanie tytułu nie jest chyba problemem.

„Zmienne systemowe" to po prostu pewne komórki w pamięci RAM Commodore 64, które są wykorzystywane przez system, czyli przez zapisane w pamięci ROM oprogramowanie fabryczne (jak np. interpreter języka Basic). Zawartość tych komórek może być modyfikowana lub po prostu odczytywana w celu uzyskania informacji związanych z aktualnym stanem komputera. Tabele z wypisanymi adresami zmiennych systemowych wraz z ich opisami były już zamieszczane wielokrotnie, tak więc nie zamierzam robić tego jeszcze raz. Dlaczego więc wspominam o tym? Niedawno mój znajomy poprosił mnie o wykonanie dla niego prostej bazy danych. Sam program nie był za bardzo skomplikowany, jedyną trudność stanowiła duża (jak na możliwości C64) ilość danych, które powinny się znajdować jednocześnie w pamięci komputera.

Po paru przymiarkach stwierdziłem, że będzie to możliwe o ile napisany przeze mnie program w języku Basic będzie zajmował jak najmniej miejsca. Wielokrotnie go zmieniając uzyskałem w końcu wersję, która bezkolizyjnie pracowała z utworzonymi z poziomu języka Basic tablicami zmiennych.

Podczas tej pracy bardzo przydatny okazał się zamieszczony obok program, który sobie właśnie na taką okazję zrobiłem.

Jego zadaniem jest nieustanna kontrola tych zmiennych, które odpowiadają za początkowe i końcowe adresy obszarów w pamięci zajętych przez program w Basic'u, zmienne i tablice oraz bieżące podawanie informacji na ekranie o pozostałej wolnej pamięci, którą można wykorzystać na rozbudowę programu lub ilości danych. Wielkość ta jest podawana w zapisie szesnastkowym i po użyciu rozkazu NEW powinna wynosić $97FD.

Program przepisuje się w nieużywane obszary pamięci na stosie mikroprocesora i modyfikuje dla własnych celów wektor przerwań maskowalnych ($0314/$0315).

Wyświetlanie informacji odbywa się na dwóch sprite'ach, które poruszają się wraz z kursorem edycji. Można w ten sposób przeprowadzić wiele interesujących eksperymentów np. definiując tablice zmiennych czy też dopisując nową linię programu z jednoczesną obserwacją zmian wielkości wolnego obszaru pamięci. Po uruchomieniu własnego programu dyrektywą RUN wyświetlanie informacji jest wyłączane.

Zamieszczone obok dane należy wpisać do komputera korzystając z dowolnego programu typu „monitor", a po dokładnym sprawdzeniu przegrać na taśmę lub dysk obszar od $0801 do $0918. Tak przygotowany program można uruchomić rozkazem RUN.

I jeszcze jedna uwaga na koniec: program korzysta z bufora magnetofonu ($0340-$03FD), tak więc aby uniknąć komplikacji przy współpracy z magnetofonem należy przerwać działanie programu poprzez wciśnięcie kombinacji RUN/STOP i RESTORE, a po skończeniu operacji z taśmą ponownie go uaktywnić poprzez SYS 272.

```
.:0801  0B 08 8E 07 9E 32 30 35
.:0809  39 00 A0 00 78 B9 26 08
.:0811  99 10 01 C8 C0 9D D0 F5
.:0819  A0 00 B9 C3 57 99 A8 02
.:0821  C8 C0 51 D0 F5 78 A9 01
.:0829  8D 15 03 A9 30 8D 14 03
.:0831  A2 7F A9 00 9D 40 03 CA
.:0839  10 FA A2 0D 8E F8 07 E8
.:0841  8E F9 07 58 60 A5 9D 30
.:0849  06 8D 15 D0 4C 31 EA A2
.:0851  FF EC 12 D0 D0 FB 8D 10
.:0859  D0 A0 03 8C 15 D0 8C 27
.:0861  D0 8C 28 D0 A9 18 A6 D3
.:0869  18 69 08 90 03 8C 10 D0
.:0871  CA 10 F5 8D 00 D0 18 69
.:0879  18 90 05 A0 02 8C 10 D0
.:0881  8D 02 D0 A9 32 A6 D6 18
.:0889  69 08 CA 10 FA 8D 01 D0
.:0891  8D 03 D0 4C A8 02 85 02
.:0899  4A 4A 4A 4A 20 A1 01 A0
.:08A1  00 84 FE 0A 26 FE 0A 26
.:08A9  FE 0A 26 FE 85 FD A5 FE
.:08B1  18 69 D0 85 FE 60 29 0F
.:08B9  C9 0A 90 03 E9 09 60 09
.:08C1  30 60 A5 34 38 E5 32 85
.:08C9  FC A5 33 38 E5 31 B0 02
.:08D1  C6 FC 85 FB A2 33 86 01
.:08D9  20 81 01 A2 02 20 EB 02
.:08E1  A5 02 20 87 01 A2 40 20
.:08E9  EB 02 A5 FC 20 81 01 A2
.:08F1  00 20 EB 02 A5 02 20 87
.:08F9  01 A2 01 20 EB 02 A9 37
.:0901  85 01 4C 31 EA B1 FD 9D
.:0909  40 03 E8 E8 E8 C8 C0 08
.:0911  D0 F3 60 00 00 00 00 00
.:0919  00 00 00 00 00 00 00 00
```

Listing programu

Paweł Sołtysiński

Programy ciekawe, zwariowane i takie sobie

Nasze pismo jest wciąż jeszcze bardzo młode. Cały czas poszukujemy własnej formy i stylu. Proponujemy coraz to nowe działy, które - mamy nadzieję - zainteresują Was i będziecie chcieli wziąć udział w ich redagowaniu.

W tym numerze rozpoczynamy rubrykę pt. „PROGRAMY CIEKAWE, ZWARIOWANE I TAKIE SOBIE". Co chcemy w niej zamieszczać? - wszystkie takie programiki, które jeszcze nie dorosły do miana PROGRAMU lecz i tak spełniają funkcje im przeznaczone. Nie muszą one być długie ani genialne. Ot, po prostu kilka linii, które robią coś śmiesznego, zawierają ciekawy pomysł, czy w niekonwencjonalny sposób rozwiązują stare problemy.

Na razie będziemy starali się wymyślać sami tematy tej rubryki. Czekamy jednak na Wasze produkty. Jeśli wymyśliliście coś ciekawego i macie ochotę podzielić się tym z naszymi czytelnikami to przyślijcie to do nas.

Pierwszy program który Wam prezentujemy jest całkowicie pozbawiony sensu. Na pomysł jego napisania wpadłem na jednym z wykładów z informatyki, podczas którego wykładowca tłumaczył zawzięcie, że zamiany dwóch zmiennych dokonujemy przez użycie trzeciej - pomocniczej. Postanowiłem oczywiście udowodnić, że można to zrobić zupełnie inaczej - nie używając żadnych zmiennych pomocniczych ani też innych rozkazów typu POKE zapamiętujących wartości w pamięci. Jak postanowiłem tak uczyniłem i oto program, który zamienia dwie zmienne A i B ze sobą:

```
10   A=8:B=3
20   A=A+B
30   B=A-B
40   A=A-B
50   PRINT A,B
```

Sprawdźcie sami! Działa!!

Historia powstania drugiego programu jest nieco odmienna, aczkolwiek połączona z tym pierwszym. Pewnego razu swojemu koledze, zapalonemu programiście w Basic'u (cześć Wojciech!) zadałem zagadkę, na którą odpowiedź Wy już znacie: „zamień dwie zmienne bez użycia trzeciej". On w zamian kazał mi napisać program na policzenie silni dowolnie wybranej liczby. Ponieważ obaj jesteśmy „wybornymi" programistami więc oczywiście zagadek nie rozwiązaliśmy. Kiedy ja podałem mu rozwiązanie mojej zagadki „walnął" się w głowę aż huknęło i powiedział: „ale to proste!". Ja uczyniłem to samo po usłyszeniu jego rozwiązania. Na czym jednak polegał problem? Sam program na liczenie silni jest bardzo prosty, podobnie zresztą jak i sama silnia.

2! (wykrzyknik oznacza właśnie silnię) równa się wyrażeniu 1*2

3!=1*2*3

8!=1*2*3*4*5*6*7*8 (przykładu dla 30! z oczywistych względów nie przytoczę)

Napisana jednak prosta pętla, która mnoży liczby jest nieskuteczna. W praktyce po podaniu wartości większej niż 34 komputer wyświetli komunikat „OVERFLOW ERROR" i liczenie na tym się zakończy. Co to oznacza? Otóż komputer też ma zakres, w którym może operować zmiennymi. Jeśli np. na kalkulatorze podacie zbyt duże liczby do np. mnożenia to uzyskacie komunikat o błędzie w postaci literki „E". To samo oznacza również nasz „OVERFLOW ERROR". Jak sobie z tym poradzić? Przeanalizujcie ten program:

```
10   INPUT „Liczba";A
20   IF A<=0 THEN 120
30   IF A/INT(A)<>1 THEN 120
40   S=1
50   FOR Q=1 TO A
60   S=S*Q:IF S>1E30THENS=S/1E20:L=L+20
70   NEXT Q
80   W=INT(LOG(S)/LOG(10))
90   IF L=Q THEN 110
100  PRINT A;" I =";S;" I I I";E+L:RUN
110  PRINT A;" I=";S:RUN
120  PRINT"?ERROR":RUN
```

Uwaga! Znak „ I" oznacza kursor w lewo, trzy znaki „ I I I" znaczą trzy razy kursor w lewo.

Czekamy na Wasze zwariowane pomysły. Najciekawsze, najśmieszniejsze, najbardziej zwariowane opublikujemykujemy. Mamy tylko jedną prośbę. Niech to będą Wasze programy!

Sambor Kuźma

P.S. Podziękowania dla Pawła Wojciechowskiego (pseudonim Wojciech) za świetną zagadkę.

Assembler 64

Prawdopodobnie każdy z posiadaczy komputera Commodore 64 spotkał się z określeniami: „język maszynowy", „kod maszynowy" czy też „assembler". Ci z Was, którzy stawiają w programowaniu procesora 6510 pierwsze kroki na pewno poznali i korzystali nieraz z monitorów pamięci z możliwością assemblacji i disassemblacji kodu maszynowego przy wykorzystaniu mnemoników rozkazów procesora i konkretnych adresów w pamięci naszego „Commodorka".

Pisząc swoje programy w ten sposób należy mieć już raczej skrystalizowany pomysł i uważać na adresy, pomiędzy którymi leży, lub do których odwołuje się pisany przez nas program.

Wyobraźmy sobie teraz, że po parogodzinnym „klepaniu" w klawiaturę decydujemy się na próbne uruchomienie naszego „dzieła" i nagle okazuje się, że gdzieś w środku jest błąd lub wręcz zapomnieliśmy wpisać kilku rozkazów! Następuje potem trudna procedura tworzenia różnych obejść, które pozwalają na naprawienie efektów naszego roztargnienia.

Niemniej jednak, przy pewnej wprawie, można w ten sposób pisać dość wygodnie niewielkie procedury, które są nam potrzebne do niezbyt ważnych zastosowań.

Jak jednak powinniśmy pisać większe programy, o dużym stopniu skomplikowania, które, być może będziemy chcieli kiedyś w łatwy sposób modyfikować? Oczywiście, w assemblerze! Bardzo wcześnie programiści wpadli na pomysł, aby przyszły program w języku maszynowym móc pisać w edytorze tekstów, a adresy zamieniać na dowolne, łatwe do zapamiętania nazwy.

Na większych maszynach assemblery to najczęściej programy, które obrabiają teksty napisane na dowolnym edytorze tekstów, natomiast na naszym 64-owym podwórku w zasadzie wszystkie assemblery posiadają wbudowane na stałe własne edytory.

Chcąc pomóc wszystkim w rozpoczęciu pracy z posiadanym (lub dostępnym w przyszłości) assemblerem pominę w wyjaśnieniach zasadę pracy z wbudowanym edytorem (w zasadzie wszystkie mi znane pracują w sposób podobny do edytora systemowego), a skupię się raczej na podstawowych zasadach pisania programu źródłowego. Jak już wspomniałem wcześniej, zamiast adresów komórek w pamięci można używać wygodnych etykiet.

I tak np. jeśli napiszemy:

```
:ETYKIETA   LDA#$00
```

to od tej pory assembler będzie kojarzył nazwę „ETYKIETA" z rzeczywistym adresem rozkazu, który po niej występuje. W ten sposób możliwe są następujące zapisy:

```
LDA  ETYKIETA
LDA ETYKIETA +3
JMP  ETYKIETA
BNE  ETYKIETA
```

i tym podobne.

Dowolnej etykiecie można nadać także jakąś wartość (stałą) np:

```
:ETYKIETA=$EA31
```

Jak już stwierdziliśmy, etykieta zastępuje dowolny adres (dwubajtowy). Podziału na młodszy i starszy bajt (np. na potrzeby jakiegoś wektora) dokonuje się przy pomocy symboli „<" i „>" (mniejsze/większe):

```
LDA #ETYKIETA  ;młodszy bajt
STA  WEKTOR
LDA  #ETYKIETA  ;starszy  bajt
STA  WEKTOR+1
```

Jak już zapewne zauważyliście, odpowiednie komentarze można umieszczać w tekście po znaku średnika (aby nie były niepotrzebnie interpretowane przez assembler).

Oprócz rozkazów assemblera i etykiet czasami trzeba umieszczać w programie różne dane, jak pojedyncze bajty czy też teksty. Do tego celu służy komenda „B" (w niektórych assemblerach „BYT"), po której wpisujemy potrzebne nam dane, np:

```
B   00,02,44,$8E,%00101100
```

lub tekst:

```
B  „tekst"
```

Na koniec ostatnia ważna rzecz: o przyszłym położeniu naszego programu w pamięci informujemy assembler za pomocą komendy „ORG" (ang. origin - początek; w niektórych assemblerach „*=").

A na koniec mały przykład programu napisanego w assemblerze. Jego analiza przyniesie na pewno wiele ciekawych wniosków.

```
ORG  $8000                    ;32768   dziesiętnie
:IRQCONT=$EA31
:ILEKOL=$0E
SEI
LDA   #$7F
STA   $DC0D
LDX   #$00
STX   $DC0E
INX
STX   $D01A
LDA   #$1B
STA   $D011
LDA   #$33
STA   $D012
LDA   #<IRQ
STA   $0314
LDA   #>IRQ
STA   $0315
CLI
RTS
:IRQ  LDX #$00
:L1   LDY TIMTAB,X
:L2   DEY
      BNE L2
LDA   KOLORY,X
STA   $D020
STA   $D021
INX
CPX   #ILEKOL
BNE   L1
INC   $D019
JMP   IRQCONT
:TIMTAB             B8,8,8,8,8,8,8,1
                    B8,8,8,8,8,8,8,1
:KOLORY             B9,2,8,10,15,7,1
                    B7,15,10,8,2,9,0
```

Paweł Sołtysiński

SPRITE edytor

W tym miesiącu nagrodą za najlepszy nadesłany na nasz konkurs program postanowiliśmy wyróżnić Marka Rzepkę ze Szczecina.

Program ten służy do edycji sprite'ów. Zajmuje obszar pamięci od adresu $4000 do $43E7, a dane lokowane są w przestrzeni od $2000 do $4000. Start programu następuje po wykonaniu komendy SYS 16574 ($40BE).

Objaśnienia funkcji:

* „+" i „-" :zmiana edytowanego sprite'a;
* klawisze kursora+SPACE :edycja na matrycy;
* F1 :włączanie i wyłączanie trybu kolorowego;

Uwaga! W trybie kolorowym jeden punkt na sprite'a równa się dwóm punktom na matrycy.

* Kombinacja:

o● - kolor nr 1;
●● - kolor nr 2;
●o - kolor nr 3;

* F3 :kasowanie sprite'a;
* 1,2,3 :zmiana kolorów w trybie kolor.

Uwaga! Jeśli nie podoba Ci się stawianie punktu, to wciśnij RESET i napisz:POKE 16962,234:POKE 16963,234:POKE 16964,234:SYS 16574.

Dla bardziej zaawansowanych podaję skróconą mapę programu:

$4189 - kolor ON/OFF
$4199 - zmiana koloru nr 1
$41A4 - zmiana koloru nr 2
$41AF - zmiana koloru nr 3
$41F8 - ruch w prawo
$4246 - ruch w lewo
$4270 - ruch w górę
$429A - ruch w dół
$4222 - postawienie punktu
$4320 - następny sprite
$436B - CLEAR MATRIX
A teraz ZERO PAGE
$33 - X
$34 - Y
$35-36 - adres na ekranie kolorów
$38-39 - adres na ekranie znaków
$3E-3F - adres aktualnie edytowanego SPRITE'a

Listing programu:

```
99 PRINT „SE ]]]]]] CZEKAJ DO 1000"
100 FOR Y=0 TO 1000: READ:POKE 16384+Y:PRINT
„S"Y:NEXT
110 PRINT „SQ OK.":FOR X=0 TO 2000:NEXT:SYS 16574
200 DATA 162, 0, 189 ,64, 64, 157, 26, 4, 189, 78, 64, 157,
66, 4, 189, 92, 64, 157, 106, 4, 190
201 DATA 106, 64, 157, 146, 4, 189, 120, 64, 157, 186, 4,
189, 134, 64, 157, 226, 4, 189, 148
202 DATA 64, 157, 10, 5, 189, 162, 64, 157, 50, 5, 189,
176, 64, 157, 90, 5, 232, 224, 14, 208
203 DATA 197, 96, 117, 65, 7, 15, 15, 14, 9, 5, 32, 19, 15,
6, 20, 48, 57, 48, 32, 32, 43, 47, 45
204 DATA 32, 58, 19, 16, 18, 9, 20, 5, 32, 32, 3, 18, 19, 18,
32, 58, 5, 4, 25, 3, 10, 1, 32, 32, 32
205 DATA 32, 43, 32, 32, 32, 19, 16, 18, 9, 20, 5, 32, 32,
19, 16, 1, 3, 5, 32, 32, 32, 32, 32, 32
206 DATA 32, 32, 32, 32, 6, 45, 49, 32, 58, 3, 15, 12, 15,
18, 32, 32, 32, 32, 32, 32, 32, 32, 32
207 DATA 15, 14, 45, 15, 6, 6, 32, 32, 32, 6, 45, 51, 32, 58,
13, 1, 20, 18, 9, 24, 32, 32, 49, 44
208 DATA 50, 44, 51, 58, 3, 15, 12, 15, 18, 19, 32, 32, 66,
229, 162, 1, 142, 16, 208, 142, 21, 208
209 DATA 202, 142, 32, 208, 142, 33, 208, 134, 62, 169, 6,
133, 55, 157, 0, 216, 157, 0, 217, 157
210 DATA 0, 218, 157, 0, 219, 232, 208, 241, 169, 128,
141, 248, 7, 169, 32, 133, 63, 141, 0, 208
211 DATA 169, 160, 141, 1, 208, 169, 16, 133, 51, 133, 52,
169, 218, 162, 144, 133, 54, 134, 53
212 DATA 32, 0, 64, 169, 41, 162, 4, 133, 48, 134, 49, 162,
0, 160, 0, 169, 46, 145, 48, 200, 192
213 DATA 24, 208, 247, 165, 48, 24, 105, 40, 144, 2, 230,
49, 133, 48, 232, 224, 21, 208, 229
214 DATA 32,196, 66, 32, 159, 255, 224, 133, 208, 3, 32,
137, 65, 224, 134, 208, 3, 32, 107, 67
215 DATA 224, 43, 208, 3, 32, 74, 67, 224, 45, 208, 3, 32,
32, 87, 224, 32, 208, 3, 32, 34, 66, 224
216 DATA 17, 208, 3, 32, 154, 66, 224, 145, 208, 3, 32,
112, 66, 224, 29, 208, 3, 32, 248, 65, 224
217 DATA 157, 208, 3, 32, 70, 66, 224, 49, 208, 3, 32, 153,
65, 224, 50, 208, 3, 32, 164, 65, 224
218 DATA 51, 208, 3, 32, 175, 65, 76, 47, 65, 173, 28, 208,
73, 1, 141, 28, 208, 32, 159, 255, 224
219 DATA 133, 240, 249, 96, 238, 37, 208, 32, 159, 255,
224, 49, 240, 249, 96, 238, 38, 208, 32
220 DATA 159, 255, 224, 50, 240, 249, 96, 238, 39, 208,
32, 159, 255, 224, 51, 240, 249, 96, 169
221 DATA 0, 162, 4, 133, 58, 133, 58, 133, 59, 134, 57,
165, 51, 24, 101, 56, 144, 2, 230, 57, 133
222 DATA 56, 166, 52, 169, 40, 24, 101, 58, 144, 2, 230,
59, 133, 58, 202, 208, 242, 165, 56, 24
223 DATA 101, 58, 144, 2, 230, 57, 133, 56, 165, 57, 24,
101, 59, 144, 2, 230, 57, 133, 57, 98
224 DATA 169, 24, 197, 51, 208, 1, 96, 160, 0, 165, 55,
145, 53, 165, 53, 24, 105, 1, 144, 2, 230
225 DATA 54, 133, 53, 177, 53,133, 55, 169, 1, 145, 53,
230, 51, 32, 159, 255, 224, 29, 240, 249
226 DATA 96, 32, 186, 65, 160, 0, 177, 56, 201, 46, 240, 7,
169, 46, 145, 56, 76, 56, 66, 169, 174
227 DATA 145, 56, 32, 145, 67, 32, 159. 255, 224, 32, 240,
249, 32, 248, 65, 96, 169, 1, 197, 51
228 DATA 208, 1, 96, 160, 0, 165, 55, 145, 53, 165, 53, 56,
233, 1, 176, 2, 198, 54, 133, 53, 177
229 DATA 53, 133, 55, 169, 1, 145, 53, 198, 51, 32, 159,
255, 224, 157, 240, 249, 96, 169, 1, 197
230 DATA 52, 208, 1, 96, 160, 0, 165, 55, 145, 53, 165, 53,
56, 233, 40, 176, 2, 198, 54, 133, 53
231 DATA 177, 53, 133, 55, 169, 1, 145 ,53, 198, 52, 32,
159, 255, 224, 145, 240, 249, 96, 169
232 DATA 21, 197, 52, 208, 1, 96, 160, 0, 165, 55, 145, 53,
165, 53, 24, 105, 40, 144, 2, 230, 54
233 DATA 133, 53, 177, 53, 133, 55, 169, 1, 145, 53, 230,
52, 32, 159, 255, 224, 17, 240, 249
234 DATA 96, 160, 0, 177, 62, 153, 0, 80, 200, 192, 63,
208, 246, 169, 41, 162, 4, 133, 64, 134
235 DATA 65, 162, 0, 160, 7, 24, 126, 0, 80, 144, 7, 169,
174, 145, 64, 76, 237, 66, 169, 46, 145
236 DATA 64, 234, 136, 16, 235, 234, 238, 246, 66, 169, 0,
201, 3, 240, 15, 165, 64, 24, 105, 8
237 DATA 144, 2, 230, 65, 133, 64, 232, 76, 218, 66, 169,
0, 141, 246, 66, 165, 64, 24, 105, 24
238 DATA 144, 2, 230, 65, 133, 64, 232, 224, 63, 208, 187,
96, 169, 128, 205, 248, 7, 208, 1, 96
239 DATA 165, 62, 56, 233, 64, 176, 2, 198, 63, 133, 62,
32, 196, 66, 206, 248, 7, 32, 159, 255
240 DATA 236, 45, 147, 240, 248, 32, 159, 255, 224, 45,
234, 240, 248, 96, 169, 255, 205, 248
241 DATA 7, 208, 1, 96, 165, 62, 24, 105, 64, 144, 2, 230,
63, 133, 62, 32, 196, 66, 238, 248,7
242 DATA 32, 159, 255, 224, 43, 240, 249, 96, 169, 41,
162, 4, 133, 48, 134, 49, 162, 0, 160, 0
243 DATA 169, 46, 145, 48, 200, 192, 24, 208, 247, 165,
48, 24, 105, 40, 144, 2, 230, 49, 133
244 DATA 48, 232, 224, 21, 208, 229, 96, 169, 41, 162, 4,
133, 60, 134, 61, 162, 0, 160, 8, 177
245 DATA 60, 201, 174, 208, 3, 58, 176, 1, 24, 126, 0, 60,
136, 18, 240, 238, 177, 67, 169, 0, 201
246 DATA 3, 240, 15, 165, 60, 24, 105, 8, 144, 2, 230, 61,
133, 60, 232, 76, 155, 67, 169, 0, 141
247 DATA 177, 67, 165, 60, 24, 105, 24, 144, 2, 230, 61,
133, 60, 232, 224, 63, 208, 193, 160
248 DATA 0, 185, 0, 80, 145, 62, 200, 192, 63, 208, 246,
96, 0, 7, 255, 240
1991 CODING BY GOONIE OF C.B.SOFTWARE
END
```

Marek Rzepka

Zapraszamy wszystkich do udziału
w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-16 i C-64. Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane). Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu. Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po **500.000** zł każda. Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma. Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

SLANG

Slang - „potoczna, niekonwencjonalna odmiana języka, nacechowana ekspresywnością, nie podlegająca ścisłym normom języka literackiego; gwara". To tyle słownikowej definicji, a jak to wygląda w rzeczywistości?

-Jo, to transfernij pod ce zero, pojedziemy to czerpakiem, a to spod romów dolinkujemy później!.

-O'k a skrin z resztą wrzucimy na kruela.

-Uhm, w stos pchnie się relokowajk, z przodu loderek i introko i kolejny hak gotowy!.

-Pewnie, co następne do kopnięcia?.

Ile razy znaleźliście się w towarzystwie koderów lub hacker'ów, mogliście usłyszeć podobne teksty. Co oni właściwie mówią? Nie są to żadne magiczne zaklęcia, ani klątwy rzucane na biednych użytkowników, lecz po prostu zwykła wymiana informacji. Jak poznać ten swoisty język? Wystarczy „podłączyć się" do towarzystwa używającego slangu lub czytać ten i kolejne numery „64 plus 4". A jak taka gwara powstaje?

Każda grupa ludzi, którzy zajmują się „nałogowo" jakimś konkretnym zajęciem np. komputerami, wykształca swój język. Tak jest na przykład z marynarzami, kierowcami, uczniami (chociaż trudno powiedzieć by zajmowali się nauką „nałogowo") czy choćby przestępcami. Jak to się dzieje? Kiedy trzeba szybko wymienić dużo informacji, która do tego musi być jeszcze bardzo precyzyjna, normalny język literacki zawodzi. Tak jest z każdym językiem na świecie, lecz język polski przoduje w tej bezużyteczności. Jeżeli kiedykolwiek tłumaczyliście cokolwiek z języka np. angielskiego, zauważyliście, że tam dwa słowa, które wyjaśniają wszystko, po polsku należałoby opisać dwoma zdaniami. Trochę to niepraktyczne, no nie? Nie można jednak całkowicie zrezygnować z naszej ukochanej polszczyzny (Pan doc. Miodek dostałby zawału). I w tym miejscu powstaje slang.

Język angielski jest językiem przodującym i w zasadzie „obowiązującym" w każdej dziedzinie techniki. Podobnie jest i z komputerami. Samo słowo komputer pochodzi od słowa „computer" zaczerpniętego oczywiście z j. angielskiego. Trudno się temu dziwić. Cały postęp techniczny (i komputerowy też) ma miejsce w krajach anglo-języcznych. Nawet jeśli tym krajem są Niemcy to i tak używa się tam terminologii angielskiej, tak zresztą jak i na całym świecie.

Jeśli chcemy więc być „bliżej świata" musimy również używać tejże terminologii. I tu dochodzimy do problemu wojny wypowiadanej przez „dyplomowanych" literatów słownictwu angielskiemu. Twierdzą oni z uporem maniaków, że zamiast słów angielskich należy wprowadzać polskie odpowiedniki specjalnie na te okazje tworzone. Panowie literaci, powiem krótko: opowiadacie bzdety! Wielokrotnie mieliśmy przykłady udanych „zastępstw" nie tylko w dziedzinie komputerowej. Dla przykładu: jak myślicie, co to jest „podgardle dziecięce"? Ot, komuś nie podobało się słowo „śliniaczek". Być może tym kimś był rzeźnik? Na pewno inne skłonności miał autor zwrotu „zwis męski" i nie były one anielskie... Ot, „zwis męski" to jest po prostu krawat. Długo się natomiast zastanawiałem nad zwrotem „zwis nocny". Co może wisieć nocy, albo w nocy? Zwis nocny to po prostu kinkiet, czyli nocna lampka wisząca na ścianie! Może się kiedyś urwała i spadła komuś na głowę?

Ale dość pastwienia się. Te zwroty nie przyjęły się . Zostały odrzucone przez naturalną selekcję, jaką przechodzi każdy wyraz. Ludzie uznali, że są niepraktyczne, śmieszne i zapomnieli o nich. Tak samo było z wieloma słowami na siłę wprowadzanymi do terminologii komputerowej. Przytoczmy choćby „sprzęg" na określenie interface'u. Kto z Was powie: „Poproszę sprzęg centronics"? Nie dość, że brzmi to śmiesznie, to jest jeszcze niepraktyczne. A nas, komputerowców, cechuje przecież praktyczność, nie? Cały świat posługuje się terminologią angielską. Wyobraźmy sobie więc, że jesteśmy we Francji i pytamy w sklepie „Can I have sprzęg centronics?". Na co sprzedawca wybałusza oczy, stuka się w czoło i groźnym dla zdrowia wykręcaniem języka stara się powtórzyć „Sss...pren....g", po czym pada zemdlony na ziemię. Jeśli więc panowie literaci macie trochę serca dla biednych komputerowców to dajcie spokój z wymyślaniem neologizmów rodem z trzeciorzędnego filmu SF. Myślę, że najtrafniej było to ujęte w jednym z „terminatorów terminologicznych" z „Komputera" (niestety nie pamiętam numeru ani autora). Oto kilka przykładów zaczerpniętych stamtąd:

myszka - przyłączny przesuwacz stołokulotoczny
joystick - drążkowy wpływacz na położenie celu
dyskietka - giętki płaskokręg informacjonośny.

Myślę więc, że jeden problem rozwiązaliśmy. Zostawmy słownictwo takie jakie jest, jakie tworzy się automatycznie w drodze naturalnej, językowej selekcji. Zostaje jeszcze problem jak te słowa zapisywać na papierze?

Tu było kilka szkół. Jedna proponowała zapis fonetyczny, czyli tak pisać jak się słyszy. W ten sposób napisany jest zresztą przykład na początku artykułu. Druga szkoła chciałaby słowa angielskie pisać po angielsku bez żadnej odmiany. A więc np. „Nie mam interface". Ostatnia - trzecia szkoła jest moim zdaniem najsłuszniejsza (wszystkie artykuły w naszym piśmie pisane są właśnie zgodnie z nią). Na czym ona polega? Ot, zachowujemy końcówki polskiej odmiany, lecz same słowa piszemy po angielsku, np. „Nie mam interface'u" lub „Nie mam joystick'a". Prawda, że wygląda to ładniej niż pierwszy i drugi sposób? Wiadomo jak wygląda słowo angielskie, jaka jest końcówka polskiej odmiany, a wszystko jest połączone zgrabnym apostrofem.

To tyle dywagacji na temat slangu. Pora by zabrać się do niego w praktyce. W następnych numerach będziemy prowadzili alfabetyczny słowniczek wszystkich wyrazów naszego slangu. Mam nadzieję, że będziecie rozumieli dzięki temu „czary" odprawiane nad maszynami przez programistów i czasem zagniecie ich podstępnym pytaniem: „Co, docyklowałeś to FLD w poziomie?".

Tymczasem pora wyjaśnić przykład naszej slangowej rozmowy z początku artykułu. Dwóch hacker'ów połamało (uczynili ją kopiowalną i pozbawili zabezpieczeń) właśnie grę. Zastanawiają się teraz jak umieścić ją w pamięci, sprężyć tak by zajmowała mniej miejsca i dołączyć swoją czołówkę. Jeden proponuje więc, by część pamięci przenieść pod adres $000, wstępnie sprężyć ją pewnym rodzajem programu kompresującego, a to co znajduje się pod pamięcią stałą ROM dołączyć później. Drugi zgadza się z nim i proponuje, by pamięć ekranu z resztą gry sprężyć na innym rodzaju programu kompresującego. Pierwszy potakuje i planuje by procedurę rozmieszczającą wszystko w pamięci umieścić w stosie. Na sam początek programu chce dać program ładujący wszystko do pamięci i intro, czyli czołówkę naszych hacker'ów, po czym stwierdza, że kolejny hack (czyli połamana gra) jest gotowy. Drugi się nie sprzeciwia i pyta co jeszcze zostało do połamania.

c.d.n.

Sambor Kuźma

Co to jest IFF?

Pierwsza prezentacja Amigi zaszokowała wszystkich oglądających jej potężnymi możliwościami graficznymi. Był to pierwszy komputer, który przeciętnemu użytkownikowi oferował możliwości o których mu się nigdy nie śniło. Taki stan rzeczy ukazał poważny problem - mnogość formatów graficznych.

Problem ten należało szybko jakoś rozwiązać, w przeciwnym wypadku komunikacja (wymiana danych) pomiędzy programami graficznymi stałaby się niemożliwa (ma to miejsce np. w przypadku komputerów IBM). Jeżeli użytkownik chciałby użyć innego programu graficznego musiałby najpierw odwołać się do programów przetwarzających grafikę z jednego formatu na drugi (konwertery).

Chcąc nie dopuścić do takiej sytuacji producenci Amigi wraz z firmą Electronic Arts stworzyli uniwersalny format graficzny - Interchange File Format (IFF).

IFF jest formatem nie tylko dla obrazków, ale może być użyty dla każdego zbioru począwszy od grafiki i animacji aż do sampli i muzyki (np. z muzyki w formacie IFF korzysta program SONIX firmy Aegis) co czyni go bardzo użytecznym i uniwersalnym.

IFF jest bardzo prostym standardem. Każdy zbiór składa się z bloków danych zwanych „chunk'ami". Jeden „chunk" składa się z trzech części:

- nagłówka - identyfikatora
- długości chunk'a w bajtach
- danych tego chunk'a

Długość chunk'a nie jest ilością bajtów w chunk'u, jak można by się spodziewać, ale względnym adresem następnego chunk'a.

Nagłówek są to cztery bajty w kodzie ASCII w zakresie od „ " (spacja) do „~" (tild). Spacje poprzedzające dane są niedopuszczalne za wyjątkiem czterech spacji jako nazwa chunk'a, który jest ignorowany.

Dla formatu graficznego wyróżniamy następujące nazwy:

FORH - początek standardowego zbioru w IFF'ie w polu danej podajemy nazwę formy:

ILBM - InterLeared BitMap - standardowy format obrazka - bitmap'y podzielone na linie i ułożone po kolei

SPRT-SPRiTe - format sprite'a

ACBM - Amiga Continous BitMap - forma używana przez Amiga Basic

ANMB - ANiMated BitMap - obrazy z animacją - forma używana np. przez Deluxe Video

ANIM - ANIMation - forma używana przez programy animujące

BMHD - BitMap HeaDer - chunk opisujący parametry obrazu. W danych występują kolejno:

UWORD (dc.w) szerokość, wysokość bitplane'ów

WORD (dc.w) pozycja x, pozycja

UBYTE (dc.b) ilość bitplane'ów

UBYTE (dc.b) znacznik maski

0 - bez maski

1 - maska razem z bitplane'ami

2 - maska jako kolor

UBYTE (dc.b) bajt kompresji

0 - nie skompresowany

1 - kompreska typu ByteRun 1

UBYTE (dc.b) bajt wypełnienia = 0

UWORD (dc.w) nr koloru, który będzie kolorem przezroczystym (tła)

UBYTE (dc.b) Xasp,Yasp - stosunek piksola

UWORD (dc.w) szerokość, wysokość obrazka

CHAP - ColorMAP - mapa kolorów zapisanych w następujący sposób: bajt wartości rejestru czerwonego, bajt zielonego i bajt niebieskiego dla każdego koloru

CAMG - Commodore AMiGa - specyficzne informacje o obrazie tylko dla Amigi: EHB (Extra Half-Bright) i HAM (Hold And Modify)

CRNG - Color RaNGe - używany np. przez Deluxe Paint'a - Chunk informacyjny

BODY - Obszar danych - bitmap'ów

ABIT - Obszar jak BODY ale dla ACBM (Bitplane'y ułożone jeden po drugim).

ANHD - ANimation HeaDer - dla programów animacyjnych.

DLTA - Tryb delta danych dla programów animacyjnych.

Dane obrazka (BODY) mogą być skompresowane (bajt copmression w BMHD).

Na dysku nr 1 „**64+4 Public Domain Pack**" znajdują się programy, które mogą być pomocne w zrozumieniu tego artykułu.

Marcin „Duddle" Dudar

Zmagania z Cooper'em cz.2

W poprzednim odcinku zmagań z Cooper'em omówiliśmy podstawowe komórki, z których będziemy korzystać przy pisaniu programów dla Coppera. Pozostało nam jeszcze kilka rejestrów do opisania. Na początku będą to rejestry:

BPL1MOD - $DFF108
BPL2MOD - $DFF10A

Rejestry te odpowiadają za modulo. Pierwszy odnosi się do ekranów nieparzystych, a drugi do parzystych. Modulo jest ilością bajtów jaką procesor wizji ma pominąć przy wyświetlaniu linii np. jeśli mamy ekran w Lo-Res'ie (tzn. 40 bajtów w linii), a modulo wynosi 10 to, jeżeli pierwsza linia zaczynała by się od adresu 0 to następna od adresu 40+10 czyli 50.

Za wymiary ekranu odpowiadają komórki:

DDFSTRT - $DFF092
DDFSTOP - $DFF094

Wartości dla tych komórek obliczamy z następujących wzorów:

Dla ekranu w trybie Lo-Res (niska rozdzielczość):

DDFSTRT=(HS/2-8,5) AND $FFF8
DDFSTOP=DDFSTRT+(ilość punktów/2-8)

HS - Położenie ekranu w poziomie

Ilość punktów - ilość punktów w jednej linii ekranowej

Dla ekranu w Hi-Res'ie (wysoka rozdzielczość) korzystamy natomiast z następujących wzorów:

DDFSTRT = (HS/2–4,5) AND $FFFC
DDFSTOP = DDFSTRT+(ilość punktów/4-8)

Przykładowo:

dla trybu Lo-Res, 320 punktów w linii i pozycji początkowej HS=$81:

DDFSTRT = $38
DDFSTOP = $D0

dla trybu Hi-Res, 640 punktów w linii i pozycji początkowej HS=$81:

DDFSTRT = $3C
DDFSTOP = $D4

Rejestrami decydującymi o położeniu ekranu, czyli o tzw. border'ach są:

DIWSTRT = $DFF08E
DIWSTOP = $DFF090

DIWSTRT: bity 15-8 wyznaczają pozycję górnego border'u, a bity 7-0 pozycję lewego.

DIWSTOP: bity 15-8 dolny border, a bity 7-0 określają prawy border.

I na dzisiaj byłby to koniec. W następnym odcinku rozpoczniemy tworzyć własne programy dla Coppera czyli tzw. Copper List'y.

Marcin Dudar

HAM?

HAM - Hold And Modify - tryb wyświetlania obrazu w Amidze, pozwalający na uzyskanie 4096 kolorów jednocześnie na ekranie w trybie Lo-Res - tj. w rozdzielczości 320x256 lub 320x512 (Interlace).

Konstruktorzy AMIGI marzyli o komputerze z ogromną ilością kolorów uzyskiwanych jednocześnie na ekranie monitora. Chcieli, aby używając stosunkowo niewielkiej ilości pamięci, uzyskać maksimum kolorów. Odrzucili pomysły o „mnożeniu" barw, czyli uzyskiwaniu innych odcieni poprzez szybką zmianę dwóch kolorów o jednakowej luminancji, ale różnej chrominancji. Częściowo udało im się zrealizować swoje marzenia. Dlaczego tylko częściowo? Otóż dlatego, że kolory nie są w pełni niezależne od siebie.

O obrazie w HAM'ie decyduje 16 kolorów zdefiniowanych przez użytkownika oraz układ bitów na poszczególnych BITPLANE'ach składających się na jeden ekran. Dla rysunków w HAM'ie musimy zdefiniować sześć BITPLANE'ów. Kolor punktu o współrzędnych X,Y będą nam definiowały bity z kolejnych BITPLANE'ów: bit 0 jest pobrany z pozycji X,Y pierwszego BITPLANE'u, bit 1 z pozycji drugiego BITPLANE'u itd. Kombinacja tych bitów definiuje nam kolor punktu. Jeżeli bity 5 i 4 są równe 00 to kolor punktu odpowiada kolorowi wyznaczonemu przez bity 3-0 z 16 zdefiniowanych przez nas barw. W przypadku gdy stan bitów 5 i 4 wynosi 01 to kolor „naszego" punktu jest kolorem punktu bezpośrednio na lewo, w którym wartości niebieskie są zastąpione wartością bitów 3-0. Dla kombinacji bitów 5 i 4 wynoszącej 10 sytuacja jest podobna z tą różnicą, że kolorem zastąpionym będzie kolor czerwony. Dla wartości tych bitów równej 11 zastąpiony będzie kolor zielony. Proste?

Duddle

D-Fast

Rozszerzenie pamięci to jedna z pierwszych rzeczy, które kupują użytkownicy Amigi.

Rozszerzenie wkładane w „expansion port" to najczęściej pamięć typu Fast. Jest ona umieszczana od adresu $C00000 - w górę w zależności od ilości. Często zachodzi potrzeba odłączyenia tej pamięci, a prawie zawsze chcemy wiedzieć jaką ilością pamięci dysponujemy. Do tego celu służą różne programy, które znaleźć możemy choćby na workbench'u jednak nie są one zbytnio „User-Friendly". Program D-Fast, którego listing prezentujemy poniżej, ma możliwość prostego wyłączania i włączania pamięci Fast, a ponadto drukuje ilość wolnej pamięci RAM, a także ilość wolnego miejsca oddzielnie dla pamięci Chip i Fast oraz największe możliwe bloki do zaalokowania (larg) w tych pamięciach. Wartości te są wyświetlane dziesiętnie lub heksadecymalnie.

Program ten ma jeszcze jedną ważną zaletę, a mianowicie to, że „chodzi" na niezależnym zadaniu. Moim zdaniem wszystkie programy pisane na Amigę powinny chodzić na własnych zadaniach, gdyż uruchamianie za pomocą komendy Run z CLI jest trochę skomplikowane, a program CShell zajmuje zbyt dużo miejsca.

Aby uruchomić program należy go dokładnie przepisać i zasemblować przy pomocy GenAm'a lub ArgAsm'a. Nie należy używać SEKI, gdyż program składa się z dwu sekcji kodu i jednej danych, a Seka kreuje tylko jedną sekcję kodu. Asemblację przeprowadzamy oczywiście na dysk.

Po uruchomieniu programu zgłasza się nam małe okienko, w którym podawane są dane o ilości pamięci oraz dwa przełączniki ON/OFF oraz HEX/DEC. Klikając myszą na przełącznik ON/OFF możemy włączać lub wyłączać pamięć Fast, a przełącznikiem HEX/DEC zmieniamy tryb wyświetlania z heksadecymalnego na dziesiętny. Wciskając prawy przycisk przy aktywnym oknie D-Fast'a możemy zamykać i otwierać jego okno. **Program D-Fast jest rozpowszechniany na trzecim dysku „64+4 Public Domain".**

D-Fast (Memory Controller)
(C) 1991 Marcin „Duddie" Dudar
Written on: 1991 13 Jan

```
Exec            = 4
FindResident    =-96
Forbid          = -132
Permit          =-138
AllocMem        =-198
AvailMem        =-216
FindName        =-276
GetMsg          =-372
ReplyMsg        =-378
WaitPort        =-384
OpenLibrary     =-552
CreateProc      =-138
Delay           =-198
MoveWindow      =-168
CloseWindow     =-72
SizeWindow      =-288
OpenWindow      =-204
Text            =-60
Move            =-240
SetAPen         =-342
SetFont         =-66
OpenFont        =-72

        section D_Fast0,code
Start   move.1      Exec,a6
move.1      $114(a6),a0
tst.1       $ac(a0)
bne.s       RunIt
lea         $5c(a0),a0
jsr         WaitPort,a6
jsr         GetMsg(a6)
move.1      d0,d7
jsr         Forbid(a6)
move.1      d7,a1
jsr         ReplyMsg(a6)
moveq       #0,d0
rts

RunIt lea   Start(pc),a1
move.1      -4(a1),d3
clr.1       -4(a1)
lea         $17a(a6),a0
lea         dosname,a1
jsr         FindName(a6)
move.1      $114(a6),a0
move.1      d0,a6
move.1      $98(a0),d1
jsr         -$60(a6)
move.1      d0,-(sp)
move.1      a6,-(sp)
move.1      4,a6
jsr         Forbid(a6)
move.1      (sp)+,a6
lea         ProcName(pc),a5
move.1      a5,d1
clr.1       d2
move.1      #$dac,d4
jsr         CreateProc(a6)
move.1      d0,a0
move.1      (sp)+,$3c(a0)
move.1      4.w,a6
jsr         Permit(a6)
moveq       #0,d0
rts

ProcName
dc.b        'D-Fast v2.0',0

section     D_Fast1,code

lea         dosname(pc),a1
moveq       #0,d0
move.1      Exec,a6
jsr         OpenLibrary(a6)
move.1      d0,dosbase
lea         intname(pc),a1
moveq       #0,d0
jsr         OpenLibrary(a6)
move.1      d0,intbase
lea         gfxname(pc),a1
moveq       #0,d0
jsr         OpenLibrary(a6)
move.1      d0,gfxbase

lea         NewWindow(PC),a0
move.1      Intbase(pc),a6
jsr         OpenWindow(a6)
move.1      d0,window
beq         Exit
bsr         InstallOwn

lea         Font(PC),a0
move.1      gfxbase(pc),a6
jsr         OpenFont(a6)
move.1      window(pc),a1
move.1      $32(a1),a1
move.1      d0,a0
jsr         SetFont(a6)

MainLoop
move.1      gfxbase(pc),a6
move.1      window(pc),a4
move.1      $32(a4),a4

moveq       #3,d0
move.1      a4,a1
jsr         SetAPen(a6)

moveq       #$32,d0
moveq       #$1e,d1
lea         CNote(pc),a0
moveq       #14,d2
move.1      a4,a1
bsr         Print

moveq       #$32,d0
moveq       #$14,d1
lea         RamNote(pc),a0
moveq       #4,d2
bsr         Print

move.w      #$b2,d0
moveq       #$1e,d1
lea         LargNote(pc),a0
moveq       #4,d2
bsr         Print

move.w      #$132,d0
moveq       #$1e,d1
lea         LargNote(pc),a0
moveq       #4,d2
bsr         Print

move.w      #$b2,d0
moveq       #$14,d1
lea         FastNote(pc),a0
moveq       #4,d2
bsr         Print

move.w      #$132,d0
moveq       #$14,d1
lea         ChipNote(pc),a0
moveq       #4,d2
bsr         Print

move.l      gfxbase(pc),a6
moveq       #1,d0
move.l      a4,a1
jsr         SetAPen(a6)

moveq       #1,d1
moveq       #$5a,d4
moveq       #$14,d5
bsr.s       Value

moveq       #4,d1
move.w      #$da,d4
moveq       #$14,d5
bsr.s       Value

move.l      #$20004,d1
move.w      #$da,d4
moveq       #$1e,d5
bsr.s       Value

moveq       #2,d1
```

```
move.w      #$15a,d4
moveq       14,d5
bsr.s       Value

move.l      #$20002,d1
move.w      #$15a,d4
moveq       #$1e,d5
bsr.s       Value

move.l      window(pc),a
move.l      $56(a0),a0
move.l      Exec,a6
jsr         GetMsg(a6)
tst.l       d0
bne.s       IsMessy
moveq       #$1e,d1
move.l      dosbase(pc),a6
sr          Delay(a6)

bra         MainLoop

Value  movem.l   d4/d5,-(sp)
move.l      Exec,a6
jsr         AvailMem(a6)
move.l      d0,d2
bsr         InsertValue
movem.l     (sp)+,d0/d1
move.l      gfxbase(pc),a6
move.l      a4,a1
jsr         Move(a6)
lea         ValueBuff(pc),a0
moveq       #8,d0
move.l      a4,a1
jsr         Text(a6)
rts

Print   movem.l   d2/a0,-(sp)
move.l      gfxbase(pc),a6
move.l      a4,a1
jsr         Move(a6)
movem.l     (sp)+,d0/a0
move.l      a4,a1
jsr         Text(a6)
rts

IsMessy move.l     d0,a1
move.l      $14(a1),d4
move.l      4,a6
jsr         ReplyMsg(a6)
cmp.l       #$100,d4
beq.s       ChangeWindow
cmp.l       #$200,d4
bne.s       IsGadget

move.l      window(pc),a0
move.l      intbase(pc),a6
jsr         CloseWindow(a6)
bsr         RemoveOwn
Exit    moveq      #0,d0
rts
IsGadget
tst.b       G1F+1
bpl.s       FastON
move.b      #-1,FastFlag
bra.s       TestBHex
FastON clr.b            FastFlag
TestBHex
tst.b       G2F+1
bpl.s       DecON
move.b      #-1,DecFlag
bra.s       GetOut
DecON clr.b             DecFlag

GetOut bra              MainLoop

ChangeWindow
move.l      Exec,a6
jsr         Forbid(a6)
move.l      window(pc),a0
cmp.w       #$14,10(a0)
bls.s       MagnifyWindow

moveq       #0,d0
move.w      #10,d1
sub.w       10(a0),d1
move.l      intbase(pc),a6
jsr         SizeWindow(a6)
move.l      Exec,a6
jsr         Permit(a6)
bra         MainLoop

MagnifyWindow
move.l      window,a0
moveq       #0,d0
move.w      6(a0),d1
neg.w       d1
move.l      intbase(pc),a6
jsr         MoveWindow(a6)
move.l      window,a0
moveq       #0,d0
moveq       #$19,d1
jsr         SizeWindow(a6)
move.l      Exec,a6
jsr         Permit(a6)
moveq       #10,d1
move.l      dosbase(pc),a6
jsr         Delay(a6)
bra         MainLoop

InstallOwn
move.l Exec,a6
move.l AllocMem+2(a6),OldAllocMem+2
move.l #NewAllocMem,AllocMem+2(a6)
move.l AvailMem+2(a6),OldAvailMem+2
move.l #NewAvailMem,AvailMem+2(a6)
rts

RemoveOwn
move.l Exec,a6
move.l OldAllocMem+2,AllocMem+2(a6)
move.l OldAvailMem+2,AvailMem+2(a6)
rts

NewAllocMem
tst.b       FastFlag
beq.s       OldAllocMem
btst        #2,d1
bne.s       NoMem
or.b        #3,d1
OldAllocMem
jmp         0.1

NewAvailMem
tst.b       FastFlag
beq.s       OldAvailMem
btst        #2,d1
bne.s       NoMem
or.b        #3,d1
OldAvailMem
jmp         0.1

NoMem moveq         #0,d0
rts

InsertValue
moveq       #7,d0
lea         ValueBuff(pc),a0
lea         DivTab(pc),a1
tst.b       DecFlag
beq.s       DEC_loop

HEX_loop
rol.l       #4,d2
move.l      d2,d1
and.w       #$f,d1
add.b       #'0',d1
cmp.b       #$3a,d1
bcs.s       HEX_ok
addq.b      #7,d1
HEX_ok move.b        d1,(a0)+
dbf         d0,HEX_loop
bsr.s       Check
move.b      #'$',-(a0)
rts

DEC_loop
moveq       #$30,d1
DEC_divide
addq.w      #1,d1
sub.l       (a1),d2
bcc.s       DEC_divide
subq.w      #1,d1
add.l       (a1),d2
move.b      d1,(a0)+
lea         4(a1),a1
dbf         d0,DEC_loop

Check   moveq        #6,d0
lea         ValueBuff(pc),a0
CHK_loop
cmp.b       #$30,(a0)
bne.s       CHK_end
move.b      #$20,(a0)+
dbf         d0,CHK_loop
CHK_end rts

window   dc.l  0
dosbase  dc.l  0
gfxbase  dc.l  0
intbase  dc.l  0

NewWindow
dc.w  $00
dc.w  0
dc.w  $1a1
dc.w  $23
dc.w  1
dc.w  0
dc.w  $340
dc.w  0
dc.w  $100E
dc.l  Gadget
dc.l  0
dc.l  Title
dc.l  0
dc.l  0
dc.w  0
dc.w  0
dc.w  0
dc.w  0
dc.w  1

Gadget dc.l Gadget2
dc.w  9,13
dc.w  32,7
GIF dc.w  $6
dc.w  $0101
dc.w  1
dc.l  GadgetRender
dc.l  SelectRender
dc.l  0
dc.l  0
dc.l  0
dc.w  1
dc.l  0

Gadget2 dc.l  0
dc.w  9,23
dc.w  32,7
G2F dc.w  $86
dc.w  $0101
dc.w  1
dc.l Gadget2Render
dc.l Select2Render
dc.l  0
dc.l  0
dc.l  0
dc.w  1
dc.l  0

GadgetRender
dc.w  0,0
dc.w  32,9
dc.w  2
dc.l  GadgetON
dc.w  $300
dc.w  0
dc.w  0

SelectRender
dc.w  0,0
dc.w  32,9
dc.w  2
dc.l  GadgetOFF
dc.w  $300
dc.w  0
dc.w  0

Gadget2Render
dc.w  0,0
dc.w  32,9
dc.w  2
```

```
dc.l  GadgetDEC
dc.w  $300
dc.w  0
dc.w  0

Select2Render
dc.w  0,0
dc.w  32,9
dc.w  2
dc.l  GadgetHEX
dc.w  $300
dc.w  0
dc.w  0


Font dc.l  topazfont
dc.l  $80001


FastFlag
dc.b  0
DecFlag
dc.b  -1


DivTab dc.1
10000000,1000000,100000,10000,1000,100
,10,1

ValueBuff
ds.b 10

topazfont
dc.b  'topaz.font',0

dosname dc.b  'dos.library',0
gfxname dc.b  'graphics.library',0
intname  dc.b   'intuition.library',0

Title dc.b  'D-Fast (Memory Controller)
v2.0! A Yeah! ',0

CNote dc.b  '(C)1991 Duddle'

RamNote dc.b  'RAM'
FastNote  dc.b  'Fast'
ChipNote dc.b  'Chip'
LargNote dc.b  'Larg'

section D_Fast2,data_chip


GadgetON
dc.l
%00001111111111000011111111111110000
dc.l
%00011100000111000111000000111000
dc.l
%00111000000011100111000000011100
dc.l
%00111000000011100111000000011100
dc.l
%00111000000011100111000000011100
dc.l
%00111000000011100111000000011100
dc.l
%00111000000011100111000000011100
dc.l
%00011100000111000111000000011100
dc.l
%00001111111111000011100000011100


ds.l  9,0

GadgetOFF
dc.l
%00111111111000111111111011111111111
dc.l
%01110000111001110000000111000000
dc.l
%11100000011101110000000111000000
dc.l
%11100000011101110000000111000000
dc.l
%11100000011101111111000011111000
dc.l
%11100000011101110000000111000000
dc.l
%11100000011101110000000111000000
dc.l
%01110000111001110000000111000000
dc.l
%00111111111000111000000011100000


ds.l  9,0

GadgetDEC
dc.l
%11111111100011111111100011111111110
dc.l
%11100011100111000000001110000111
dc.l
%11100001110111000000001110000000
dc.l
%11100001110111000000001110000000
dc.l
%11100001110111111000011100000000
dc.l
%11100001110111000000001110000000
dc.l
%11100001110111000000001110000000
dc.l
%11100011100111000000001110000111
dc.l
%11111111100011111111100011111111110


ds.l  9,0

GadgetHEX
dc.l
%11100001110111111111101110000111
dc.l
%11100001110111000000001110000111
dc.l
%11100001110111000000001110001110
dc.l
%11100001110111000000000111011100
dc.l
%11111111110111111000000011111000
dc.l
%11100001110111000000000111011100
dc.l
%11100001110111000000001110001110
dc.l
%11100001110111000000001110000111
dc.l
%11100001110111111111101110000111


ds.l  9,0

END
```

Marcin „Duddie" Dudar

IntuiTracker v.1.23

IntuiTracker to jeden z programów znajdujących się na naszym pierwszym dysku PD. *Do czego on służy? Otóż jest to bardzo wygodna w użyciu „odtwarzaczka" modułów napisanych pod NoiseTracker'em. Największą jej zaletą jest możliwość słuchania modułów wcześniej sprężonych Power Packer'em. IntuiTracker po załadowaniu takiej muzyczki najpierw ją depakuje, a potem zaczyna grać. Dzięki temu użytkownik może oszczędzić wiele miejsca na dyskach. Na tym samym PDP znajduje się też* **Power Packer-** *macie więc możliwość przechowywania swoich ulubionych muzyczek w wygodniejszej, sprężonej formie.*

Oto krótki opis możliwości IntuiTracker'a.

Wszystkie opcje są uruchamiane z „gzymsu" czyli przez naciśnięcie prawego przycisku myszki lub przez kombinacje klawisza „A" z innymi przyciskami. Z okna „Control" można startować i zatrzymywać odtwarzanie muzyki (START i STOP), a także włączać tryb „programowalny" (PROGRAM MODE). Dzięki temu możemy kazać naszej Amidze grać po kolei wybrane przez nas melodie. Opcja „Leave song" pozostawia melodię i wychodzi z programu.

Oryginalne moduły muszą być umieszczone w katalogu „modules", lecz można to zmienić. „Choose disk" w oknie „Tracks" wczytuje do pamięci komputera wszystkie dostępne melodie znajdujące się na dyskietce. Po użyciu tej opcji ponowne wejście do „Tracks" spowoduje wyświetlenie nazw znalezionych modułów. Jeśli moduły znajdują się w innym katalogu niż „Modules" to po „Choose disk" należy podać nazwę naszej szuflady.

Ostatnim oknem jest „Prefs", które pozwala nam między innymi na włączenie graficznego przedstawienia większości funkcji programu (Control panel). Możliwa jest również aktywacja equalizera i „suwaka" głośności. Program może również korzystać z tzw. request'era jednak wtedy konieczna jest obecność arp.library na dysku. Jeśli nazwa naszego modułu muzycznego nie zaczyna się od liter „MOD." to należy wyłączyć opcję „MOD. files only".

W następnych wydaniach naszego PUBLIC DOMAIN PACK będziemy starali się sukcesywnie przedstawiać inne światowe osiągnięcia muzyków. Zachęcamy również do przesyłania nam Waszych produktów. Będziemy je rozprowadzać na dyskach i kto wie... może dzięki temu zdołamy odkryć prawdziwe talenty! A ponieważ dobrzy muzycy są bardzo poszukiwani, więc jest to dla Was duża szansa!

Sambor Kuźma

Indiana Jones And The Last Crusade

Dziś proponujemy Wam przeżycie jednej z najlepszych gier przygodowych. Z góry uprzedzam, że nie jest to tzw. „walk through" czyli opisanie gry krok po kroku. Jeśli graliście już w tę grę i utknęliście w którymś momencie to moje sugestie na pewno pomogą Wam ruszyć dalej.

Uwaga właściciele Amig z 0,5 Mb - będziecie potrzebować bardzo dużo cierpliwości - ciągłe przekładanie dysków może doprowadzić do szału !

Uff... nareszcie w domu !. Chociaż cały mokry, jesteś w świetnym nastroju. Udało Ci się odzyskać „Cross of Coronado". Szybko przebierasz się i jakby przeczuwając dalsze kłopoty idziesz potrenować boks. Jakiś głos Ci podpowiada, że musisz poprawić lewy prosty, bo od tego może zależeć twoje życie !. Po treningu idziesz do swojej pracowni. Oj ! Ci wstrętni studenci ! Może tak Irena zajęłaby się nimi ? Podczas twojej nieobecności dostałeś wiele listów od twoich wielbicielek. Tak szczerze, to mało cię one obchodzą, więc grzebiesz w nich szukając czegoś ciekawszego. Być może gdzieś tam jest paczka od Twojego ojca ? Ponowne spotkanie ze studentami nie jest dla Ciebie zbyt miłe, więc wychodzisz oknem.

Zaproszenie pokazane Ci przez dwóch smutnych panów jest dość ciekawe (Browning to był czy Smith and Wesson ?) więc bez większych oporów udajesz się z nimi.

Donoran to straszny smutas, lecz zdołał Cię zainteresować swoją opowieścią. Okazuje się, że Twój ojciec zaginął w bliżej niewyjaśnionych okolicznościach. Cóż, trzeba będzie znaleźć najpierw jego, a potem Świętego Grala, o którym to tak uparcie marudził Donoran. Jedziesz więc do domu ojca i ... niedość, że go tam nie ma, to jeszcze jakiś barbarzyńca przewrócił pokój do góry nogami. W zdenerwowaniu przewracasz jedyny jeszcze stojący regał i ... co to ? Kluczyk owinięty w taśmę! Czym by tu go otworzyć? Może kwasem? Tak! Przecież w pracowni masz go trochę w słoiku. Kluczyki na ogół otwierają zamki więc wyciągasz starą książkę, która w rzeczywistości jest imitacją dziennika twojego taty i zabierasz obrazek, który wiele lat temu narysowałeś w prezencie dla ojca. Teraz gdzie? Do Wenecji!

Biblioteka jest ogromna, jest w niej wiele ciekawych książek a wśród nich trzy, które Ciebie interesują najbardziej. Czerwona szarfa też się do czegoś przyda, podobnie jak i słupek na którym jest zawieszona. Zaglądasz do dziennika taty i spostrzegasz, że okna w bibliotece są podobne do tych, które narysował Twój ojciec. Hmm... podobne, ale nie takie same! Szukasz więc odpowiednie-

go pomieszczenia i przypominasz sobie notatkę taty: „Pierwszy numer po prawej". Nie, może to było„Drugi numer po lewej"? Tak! lewa strona to tam gdzie jest lewa ręka, a prawa strona to tam gdzie prawa ręka. Na małych tabliczkach są małe rzymskie numerki. Bierzesz więc metalowy słupek i walisz... w duży rzymski numerek!

Stęchłe powietrze drażni Twój nos, ale takie jest życie Indiana Jones'a. Podziemia nie są jednak dla ciebie straszne bo przecież masz mapę! (prawda, że ją zabrałeś?!). Szkielet starego pirata będzie Ci chyba niezbyt przydatny, ale hak - kto wie? Idziesz dalej i widzisz klapę z drabiną. Kiedyś już tu byłeś, prawda? Facetowi tłumaczysz, że wino które pije jest beznadziejne i zabierasz butelkę. Butelki powinny być pełne więc nabierasz do niej wody i wracasz do podziemi.

Może to wydać Ci się dziwne, ale czasami nawet latarnie potrzebują trochę wody (zwłaszcza wtedy gdy są zaklejone błotem).

Oj, upadek! Na szczęście był niegroźny i możesz kontynuować wędrówkę. Kapiącej wody się nie boisz i znajdujesz stare inskrypcje na ścianie. Lepiej wbij je sobie w głowę! Drewniany korek u góry jaskini zastanawia Cię. Jak go wyciągnąć? Pejczem! Tylko trzeba go za coś zaczepić - może za... Jasne! Woda wylewa się ogromnym strumieniem a Ty wracasz po drabinie do miejsca, które kiedyś wypełnione było wodą. Idziesz dalej i widzisz maszynę. Naprawa nie sprawia Ci zbyt wiele kłopotu i chociaż nie wiesz po co, na wszelki wypadek zakręcasz kołem. Otworzenie kolejnego przejścia nie sprawia Ci kłopotu bo przecież masz dziennik ojca! Z następnym jest gorzej, ale twój wrodzony talent muzyczny daje sobie radę i tym razem. Trumny zawsze bardzo Cię interesowały, więc przyglądasz się jej uważnie i wyruszasz do zamku BRUNWALD.

Kamerdyner nie stanowi dla Ciebie problemu i już spokojnie możesz penetrować zamek. Spokojnie?! Gdyby jeszcze te kilka batalionów żołnierzy poszło na przepustkę...! Pierwszy żołnierz na pewno myśli, że przyszedłeś porozmawiać z więźniem. Poza tym wygląda na strasznego gadułę więc przestraszy się gdy mu powiesz, że zdradza ściśle tajne informacje. Uniform służącego będzie Ci bardzo pomocny, a drugiemu naziście bardzo podoba się twoja marynarka. W skrzyniach na ogół leżą pieniądze, których oczywiście nikt nie pilnuje. Kolejny Szwab myśli, że jesteś służącym i że przyniosłeś rysunek dla jego szefa. W kolejnej skrzyni znajdujesz uniform z małym kluczykiem. Jak już wiesz kluczyki otwierają zameczki, więc z szarym, niemieckim mundurem pod pachą idziesz dalej. Pijanemu żołnierzowi tłumaczysz, że kufel już nie będzie mu potrzebny. Gorące węgle tak parzą, że przydałoby się je czymś zalać. Mięso wygląda apetycznie, ale na razie nie jesteś głodny. Napełniasz jeszcze raz kufel i idziesz na drugie piętro zamku. Po drodze załatwiasz jeszcze jednego faszystę i opatrujesz swoje rany. Biedny, wystraszony chłopaczek na pewno będzie się ciebie bał, więc już jako faszysta wymyślasz mu za brudny mundur. Zdajesz sobie sprawę z tego, że piesek jest bardzo głodny i przestanie pilnować szafek kiedy dasz mu jeść. Trafiasz w końcu do centralki systemu alarmowego zamku. Podpowiem Ci, że ten żołnierz kocha czytać dzieła Hitlera i że trochę piwa na pewno zaszkodzi każdemu urządzeniu elektrycznemu. W tak zwanej „galerii malarstwa" zastanawia Cię krzywo powieszony portret Mona Lisy. Otworzenie sejfu nie sprawia Ci już kłopotu i możesz dokładnie przyjrzeć się rysunkowi Świętego Grala. Znowu trafiasz na drugie piętro i zdając sobie sprawę z tego, że odpowiednia ilość „krwi w alkoholu" osłabia refleks upijasz Biff'a. Pamiętaj jednak, że Biff jest duży i musi dostać odpowiednio dużą działkę z odpowiednio dużego naczynia!. Znalezienie srebrnego kluczyka nie przedstawia wielkiego problemu i już uratowaliśmy ojca !!!.

Kolejny ruch jest niezbyt przyjemny. Musisz pójść na całkowitą współpracę z Niemcami. Nie martw się! Jeszcze ich wykołujesz!.

Zostaliście razem z ojcem związani, lecz przecież każdą linę można przeciąć czymś ostrym. Jedyną ostrą rzeczą w pokoju jest halabarda stojąca razem ze zbroją. Lepiej dobrze wymierz!

Kominek wydaje Ci się jakiś dziwny i już jesteś w Berlinie! Autograf Hitlera w paszporcie będzie bardzo pomocny. W porcie lotniczym kradniesz samolot i lecisz do Iskanderun (tam jest ukryty Św. Gral). Po katastrofie kradniesz samochód i spokojnie mijasz wszystkie posterunki.

Teraz czekają Ciebie trzy trudne próby. W pierwszej wystarczy kliknąć myszą pod nogami Twojego poprzednika. Druga jest bardzo prosta, a w trzeciej po prostu nie wolno się zastanawiać tylko trzeba iść!

Wybór właściwego Gral'a jest dość łatwy, więc po uleczeniu ojca zwracasz go rycerzowi. I w ten sposób skończyłeś grę.

Szczególne podziękowania należą się Markowi Stawnemu i Grzegorzowi Mikule, bez pomocy których chyba nigdy bym tej gry nie ukończył.

Hi-Man

Public Domain

Przypuszczamy, że wielu czytelników zetknęło się już z tym terminem. Wyjaśnijmy jednak, że określenie Public Domain można przetłumaczyć jako „własność publiczna". Nazwę tę nadano dużej grupie programów, które nie są czyjąś konkretną własnością, a więc wszyscy użytkownicy mogą je posiadać i z nich korzystać bez obawy o ich nielegalne pochodzenie itp.

Najczęściej spotykanymi przykładami „Public Domain" są programy demonstracyjne (tzw. dema) i programy użytkowe, które zostały przez autorów oddane w ręce użytkowników na zasadzie własności publicznej.

Wzorem renomowanych magazynów komputerowych również i my postanowiliśmy umożliwić czytelnikom „64 plus 4"dostęp do najświeższych pozycji Puplic Domain z kraju i ze świata. Tak więc co miesiąc będzie można otrzymać w naszej redakcji **„64 plus 4 PUBLIC DOMAIN PACK"!**

Począwszy od stycznia czeka na Was zestaw świeżych programów demonstracyjnych i użytkowych dla C-64 i AMIGI. Oprócz programów Public Domain na dyskietkach zamieszczać będziemy programy, których listingi prezentujemy w naszym piśmie oraz ciekawsze prace nadesłane na nasze konkursy. W zestawach umieszczać będziemy również nadesłane nam programy, które z racji np. swej długości (m. in. własne programy demonstracyjne) nie będą mogły startować w naszym konkursie na najlepszy program miesiąca.

Dyskietki z Public Domain będą przygotowywane tylko dla komputerów C-64 i AMIGA. Rodzina C-16, jak na razie, nie może poszczycić się regularnym dopływem odpowiedniej ilości oprogramowania. Oczywiście nie mamy tu na myśli gier, które najczęściej własnością publiczną nie są.

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" należy zamawiać drogą pocztową wpłacając na konto bankowe redakcji kwoty:

- **20.000 zł za „64 plus 4 PUBLIC DOMAIN PACK - C-64",**
- **25.000 zł za „64 plus 4 PUBLIC DOMAIN PACK - A".**

Kwoty te obejmują koszt dyskietki (dla C-64 dysk 5.25", dla Amigi 3.5"), oraz koszty kopiowania, opakowania i przesyłki pocztowej. Chcąc zapewnić sobie stałe otrzymywanie naszych „Pack'ów" można wykupić prenumeratę. Przyjmujemy ją na podstawie wpłaty dokonanej na konto:

Bank PKO SA Oddział w Bydgoszczy, konto nr: 5.09011-400522.7-136-11-111.0.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: imię i nazwisko (lub nazwę instytucji), dokładny adres zamawiającego, skrót „PDP-64" (jeśli zamawiamy zestaw dla C-64) lub „PDP-A" (jeśli zamawiamy zestaw dla Amigi). Cena jednego egzemplarza PDP-64 w prenumeracie wynosi 18.000 zł, (12 numerów - 216 tys. zł) a PDP-A 22.000 zł (12 numerów - 264 tys. zł). Prenumeratę można zawrzeć w dowolnym miesiącu na okres od trzech do dwunastu miesięcy (do końca roku kalendarzowego). **Prenumerata może obejmować miesiące od początku roku-tzn. zamawiając prenumeratę na cały rok np. w marcu, w pierwszej przesyłce otrzymacie wszystkie poprzednie zestawy (tj.styczniowy i lutowy).** Wśród pierwszych stu prenumeratorów rozlosowane będą atrakcyjne nagrody! Planujemy również wielkie głosowanie (i wielkie nagrody) na najlepsze demo z naszych „Pack'ów".

NAUKA I ŚWIETNA ZABAWA, CO MIESIĄC NOWOŚCI - TO WSZYSTKO ZAPEWNI CI NASZ „64 PLUS 4 PUBLIC DOMAIN PACK" !!

Redakcja

P.S. Spis zawartości naszych zestawów na stronie 21.